Beköszöntő

Márciusi számunk anyagának jelentős része a hálózati alkalmazásokkal foglalkozik.
Szó esik például a virtuális magánhálózatokról, a Kerberos segítségével megvalósítható központi hitelesítésről, a merevlemez nélküli hálózati ügyfelekről, illetve arról, hogyan valósíthatunk meg Linux segítségével biztonságos FTP-kiszolgálót. Ez utóbbi témával kapcsolatban egy új név olvasható a cikk végén. Van szerencsénk bemutatni ***Maróti Tamást***, aki remélhetőleg a jövőben is gyakran szerepel majd a stáblistán.
Hasonlóan örvendetes, hogy ebben a lapszámban ismét túlsúlyba kerültek a magyar szerzőink által írt cikkek. Lehet, hogy jön a tavasz?
Mint az köztudott, a ***Novell*** nemrég felvásárolta a ***SuSE Linuxot***, ami feltehetőleg jelentős változásokat hoz majd ennek a terjesztésnek a fejlődésében. Éppen ezért tartottuk fontosnak, hogy rögtön beszámoljunk a fejleményekről.
A merevlemez nélkül működő X terminálok témaköre az átlagos otthoni felhasználónak talán nem különösebben fontos, ugyanakkor az iskoláknak, illetve a ***Linux*** üzleti felhasználóinak nagy segítséget, és főleg jelentős költségmegtakarítást jelenthet ***Chip Coldwell*** rövid bevezetője.
Ugyanezen a felhasználási területen, vagyis a nagyobb rendszerezek tulajdonosai és üzemeltetői körében tarthat számot érdeklődésre ***Dr. Alf Wachsmann Kerberos 5***-ről szóló cikke, illetve ***Mick Bauer*** írása, aki ebben a hónapban a virtuális magánhálózatok lelkivilágát boncolgatja.
Magyar szerzőink közül ***Horváth Ernő*** a ***Moodle*** nevű elektronikus oktatórendszer használatát, illetve a hagyományos oktatási modellbe való beilleszthetőségét tárgyalja.
Tavaly decemberi számunk mellékletén mutattuk be az ***Ubuntu Linuxot***, amely egy viszonylag új, többnyelvű terjesztés. ***Csontos Gyula*** ennek a rendszernek a telepíthető változatát próbálta ki.
Beszédes Balázsnak a felhasználói viselkedés megfigyeléséről szóló ötödik cikkével végetér a Digitális Nagy Testvér módszereit bemutató sorozatunk, de reméljük nem kell sokat várnunk egy újabb, hasonlóan érdekes témára.
A ***Reflection API*** bemutatásával ugyancsak befejeződik ***Komáromi Zoltánnak*** a ***PHP 5*** újdonságait tárgyaló sorozata, ***Illés Viktor*** ellenben még csak most kezd belelendülni a ***Samba 3*** képességeinek bemutatásába. Végül ***Garzó András*** sorozatának újabb részében a hálózati forgalomirányítás elveit mutatja be.

Jó szórakozást kíván
a Linuxvilág stábja!

Ide tessék!

Különös, elektronikus jelzőtáblákat mutatott be a *Lucid*. A szállodákba, múzeumokba, éttermekbe szánt táblák egy 17"-os, 1280x1024 képpont felbontású, álló és fekvő helyzetbe is fordítható *LCD TFT* kijelzőt tartalmaznak, erre lehet kinyomtatni a vendégekkel közölni kívánt információkat.

A nyomtatás szó használata sem véletlen, ugyanis a jelzőtáblákat nyomtatóként kezelheti a felhasználó, aki bármilyen programmal előállíthatja a kívánt oldalt, majd átküldheti azt a jelzőtáblára. A táblák beállításai webes felületen keresztül adhatók meg, és képesek arra is, hogy a nyomtatási sorukban lévő oldalakat egymás után ismételve jelenítsék meg. Memóriájuk 256 MB méretű, ami átlagosan 50 oldal tárolására elegendő.
A cég multimédiás táblák fejlesztésén is dolgozik, ezek a jelenlegieknél gazdagabb tartalom megjelenítésére lesznek képesek; valamint rendelkezik egyéb hasonló, az épületek folyosóira, ajtóira kihelyezhető, a vendégek útba igazítására használható termékekkel is. A kijelzőkben egy *Cirrus ARM* processzor található, mely a *Linux* rendszermag 2.4.21-es változatát futtatja.
➲ www.lucid.com

Osztódás és evolúció

A *Free Standards Group* úgy döntött, több modulra osztja fel a *Linux Standards Base (LSB)* előírás-gyűjteményt – a várakozások szerint így a gyártók könnyebben tudnak majd a szabványoknak megfelelő asztali vagy kiszolgáló operációs rendszereket összeállítani. A modulok révén a jelenleginél sokkal többféle területet lehet majd szabványokkal lefedni, ugyanakkor a gyártóknak sem kell majd például egy kiszolgáló szerepre fejlesztett operációs rendszerrel inkább asztali gépeknél elvárt előírásokat teljesíteniük; illetve maguk választják ki, hogy a kötelezően betartandó fő előírások mellett mely további, elhagyható szabályokat kívánnak figyelembe venni. Várhatóan készülni fog egy szabványsablon is, amely a különféle futtatási környezetek – elsőként a Java – szabványosítására lehet majd használni. Az *LSB* szabvány iránt egyébként egyre nagyobb az igény, főleg Ázsiából, ahol a *Linux* inkább csak az utóbbi időben kezdett el terjedni. Az *LSB* soron következő, 3.0-s változata még ebben a negyedévben elkészül, következő kiadásának megjelentetését pedig a jövő évre tervezik.
➲ www.freestandards.org

MP3-biznisz kicsit másképp

Michael Robertson, a még 1997-ben indított *MP3.com* webhely egykori gazdája ismét zenei oldal indítását tervezi.

Robertson – mellesleg a *Linspire Inc.* alapítója – akár fityiszt is mutathatna az egyre kifinomultabb másolásvédelmi és digitális jogkezelő megoldásokat alkalmazó, már meglévő zeneárusító oldalaknak, ugyanis új, *MP3Tunes.com* címre kerülő oldalán mindenféle védelem nélküli *MP3* fájlok vásárlására nyílik majd lehetőség. Az *MP3Tunes.com* az egyes számokat 88 centért, a teljes albumokat pedig 8,88 dollárért kínálja majd, vagyis versenyképes árat biztosít vevőinek. A másolásvédelem hiánya természetesen a nagyobb kiadókat elriasztja, ám kisebb kiadók és független előadók számaiból így is összegyűlt az új oldalra mintegy kétszázezer, ami induló árukészletnek nem is rossz. A vevők a fizetés után korlátozás nélkül, tetszőleges lejátszókészülékre átmásolhatják a zenéket, illetve CD-lemezre írhatják őket.
Robertson szerint a vásárlókat idegesíti, hogy az általuk megvásárolt zenéket csak egy-egy készüléken tudják meghallgatni, és azt is abszurdnak tartja, hogy akik kalózmásolatokat használnak, semmilyen korlátozással nem kénytelenek számolni, a jogszerű felhasználók viszont folyamatosan korlátokba ütköznek. A gondolatmenet kétségkívül logikus, ám független felmérések szerint a vásárlók túlnyomó részét a legkevésbé sem érdeklik a korlátok és a védelmi megoldások, amíg a saját készülékükön gond nélkül tudják hallgatni a megvásárolt zenéket. Az igazság valószínűleg ismét valahol a két álláspont között van.
➲ http://mp3tunes.com

Nyitás a világra

Immár nyílt forrással, *GPL* szerződés hatálya alatt is elérhető a *Linux Canada Quasar* nevű, komoly tudású számlázóprogramja. Az 1.4-es változatnál tartó alkalmazás megnyitásának két fő oka az, hogy a világszerte eltérő és változó igényeket egy kisebb, zárt forrással dolgozó fejlesztői csapat képtelen lenne követni, illetve a vállalkozások egyre nagyobb figyelmet fordítanak az általuk használt alkalmazások forráskódjának elérhetőségére, megismerhetőségére. A nyitás révén a jelenleg inkább csak az angol nyelvű területeken ismert *Quasar* valószínűleg a világ további részein válhat ismertté – amint a lelkes önkéntesek elkészítik a helyi előírásoknak megfelelő formátumokat kezelő és persze honosított változatokat. A cég megtartja hagyományos konstrukcióit is, vagyis aki igényli, az fizetés ellenében támogatási szolgáltatással is hozzájuthat a programhoz.
➲ www.linuxcanada.com

Végleges linuxos SkyPE

Mac OS X és *Linux* alá is elkészült a *SkyPE*, a világ legnépszerűbb *VoIP*-ügyfelének 1.0-s változata. Ezzel megkettőződött a *SkyPE* futtatására alkalmas operációs rendszerek száma, eddig ugyanis csak *Windowsra* és *Pocket PC*-re létezett az alkalmazás. A tesztváltozatok még nyár közepén jelentek meg, a mostani kiadással ezzel véglegesítésére került sor. A *SkyPE*-t világszerte több mint 23 millióan használják internetes és ezzel gyakorlatilag ingyenes telefonhívások lebonyolítására. Természetesen a *SkyPE* szolgáltatásai ennyiben nem merülnek ki, a felhasználóknak lehetőségük van az internetes világból való kilépésre is, így nemcsak *VoIP* végpontokat érhetnek el.

A jövőbeli fejlesztések további operációs rendszerek támogatását, vezeték nélküli okos telefonok hálózatba vonását és értéknövelt szolgáltatások – például hangposta – biztosítását célozzák.
➲ www.skype.com

Pályaváltás

Mérföldkőhöz érkezett a német vasúttársaság, a *Deutsche Bahn Linuxra* való átállást célzó tervezete. A *DB Lotus Notes* rendszerét állította át sikeresen *UNIX* alapokról a *SuSE Linux Enterprise Server 8*-as változatát futtató *IBM eServer zSeries 990*-es gépekre. A több lépéses, az *IBM* aktív közreműködésével végrehajtandó átállás végére a *DB* a várakozások szerint olcsóbb, gyártófüggetlenségénél fogva rugalmasabb és magasabb fokú együttműködési lehetőségeket kínáló rendszerhez fog jutni. A cég 2005 végére fejezi be az áttelepítéseket, addigra *SAP* alapú vállalatirányító rendszerétől kezdve az adatbázisokon és a webkiszolgálókon keresztül egészen az utastájékoztató rendszerig minden linuxos alapokra kerül. A mostani átállás méreteit sem szabad lebecsülni: a levelezőrendszer 55 ezer felhasználót szolgál ki, és összesen 5500 adatbázisban 6,5 TB-nyi adatot tárol.

Megújult Centrino

Az *Intel* megkezdte a *Centrino* mobil platform következő generációját alkotó termékeinek szállítását. A *Sonoma* nevű együttes elődjeihez hasonlóan mobil processzort, lapkakészletet és vezeték nélküli hálózati csatolót foglal magába; a processzorok órajele 1,6 és 2,13 GHz közötti, de a termékvonal egy alacsony feszültségű, 1,5 GHz órajelű, valamint egy ultra-alacsony feszültségű, 1,2 GHz-es processzort is magába foglal. A *Sonoma* lapkakészlete az *Alviso*, mely 533 MHz-es előoldali busszal üzemel, támogatja a *DDR* memóriák kétcsatornás kezelését, a *PCI Express* szabványt, valamint tartalmazza az Intel Azalia nevű, 7.1-es *Dolby* hangzást biztosító hangkártyáját is. Ugyancsak fontos fejlemény, hogy az *Intel* a *Linux* rendszermagon végrehajtott, a 2.6.8-as kiadással elérhetővé vált fejlesztések nyomán engedélyezte a gyártóknak, hogy *Linuxot* futtató gépeiken is feltüntessék a *Centrino* logót. A *Linux* ugyan eddig is futott a centrinos gépeken, ám energiagazdálkodási lehetőségei, és ebből fakadóan a linuxos gépek akkumulátoros üzemidői nem feleltek meg az *Intel* elvárásainak, így ezeket a gépeket csak a marketing szempontból fontos jelölés nélkül lehetett értékesíteni.
➲ http://intel.com/products/notebook/centrino/index.htm

Lemondó nyilatkozat

Roppant rokonszenves újévi ajándékkal kedveskedett az *IBM* a nyílt forrású programok fejlesztői közösségének: 500 az *Amerikai Egyesült Államokban* és más országokban bejegyzett szabadalmát tette számukra szabadon felhasználhatóvá. A használati jogok átadása minden egyénre, közösségre vagy vállalatra érvényes, aki vagy amely az *Open Source Initiative (OSI)* kezdeményezés irányelveinek megfelelően végzi programfejlesztői tevékenységét. A szabadon hozzáférhetővé vált szabadalmak a számítástechnika legkülönbözőbb területeire vonatkoznak: tárolókezelés, többprocesszoros adatfeldolgozás, ember-gép kapcsolat, adatbázisok, képfeldolgozás, titkosítás és tömörítés, hálózatkezelés, internetes megoldások stb. Természetesen szó sincs arról, hogy az *IBM* közprédává tenné szellemi tulajdonát, hiszen az általa bejegyeztetett szabadalmaknak csak egy kis töredékéről van szó – a cég nagy szabadalomrekorder, az elmúlt években évente több mint 3000 bejegyzett újítást mondhatott a magáénak. A nyitás célja az, hogy a szabványos, egymással együttműködni képes rendszerek fejlesztését ne akadályozzák a szabadalmak; miközben a nyílt forrású megoldások fejlődése nyilvánvalóan a szolgáltatások felé elmozduló *IBM*-nek is kedvez.
➲ www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf

Nem kell a buta PDA

Felmérések szerint immár harmadik éve folyamatosan szűkül a személyi digitális asszisztensek, röviden *PDA*-k piaca. A vásárlók elsősorban a telefonkészülék nélküli, jellemzően alapszolgáltatásokat biztosító készülékektől fordulnak el, a gyártók a tavalyi évben öt éve először már tízmillió ilyen titkárt sem tudtak eladni. Érdemes ugyanakkor figyelembe venni, hogy bizonyos gyártók – *Windows* operációs rendszert futtató – zsebtitkárainak piaca bővült, vagyis az egyszerű adatkezelésnél többre is képes készülékek iránt folyamatosan növekedik az igény.

Medgyesi Zoltán
(mz@rettesoft.hu)
A Linuxvilág hírszerkesztője. Szabadidejét legszívesebben a barátnőjével tölti, szeret autózni és bográcsban főzni.

Mi újság a rendszermag fejlesztése körül?

Pavel Machek felfedezett egy igen komoly hibát a 2.4-es rendszermagban: 9223372034708485227 január elsején minden 2.4-es rendszer parancsfeldolgozása leáll. Igaz néhány tudós azt állítja, hogy nem érdemes olyan eseményekkel foglalkozni amelyek kilenctrillió év múlva következnek be, éppen ez a hozzáállás vezetett a *Y2K* hibához az *ioctl* csatolófelületben. Nekünk, törődő Linuxosoknak figyelnünk kell az ilyen részletekre, amely felett az üzleti világ elsiklik. Ha most kijavítjuk a hibát, nagyon jó hálózati rendszermag folt lesz belőle; és ha várunk kilenctrillió évet, ki tudja hány számítógép fut majd ezzel a folttal a galaxis távoli zugaiban? Nem beszélve róla, hogy leszármazottaink valószínűleg szorosan egybeépülnek majd számítógéprendszereikkel. '27 január elsején valamennyien hirtelen abbahagyják a parancsok végrehajtását, aztán némán és élettelenül ülnek a idők végezetéig. Hacsak nem cselekszünk most! *Marcelo*, ne vesd el ezt a foltot! Minden élet jövője függhet most tőled!

Andries Brouwer, aki a *man*, *util-linux* és *kbd* programokat, valamint a rendszermag *man* oldalakat kezelte több mint tíz éve belefáradt a munkába. Mivel ezeknek a vénséges és fontos projekteknek folytatódniuk kell, és *Andries* más vonalon szeretne továbbhaladni kérte, hogy valaki lépjen a helyére. A súgóoldalakat *Michael Kerrisk* vette át, de a többi eszköz továbbra is a levegőben lóg. Amíg megfelelő helyettest talál, kétségtelenül folytatni fogja a programok frissítéseinek kiadását, de akiket esetleg érdekelne valamelyik projekt átvétele, nézzenek utána a weben, hogy szükség van-e még helyettesre és ha igen, lépjenek kapcsolatba vele.

Ed Schouten megtette az első lépést a *Linux Xbox*-ra átvitele terén: létrehozta a konfigurációs kapcsolót hozzá. Ugyan az *Xbox* felhasználók már igencsak éheznek egy *Linux* változatra, néhány fejlesztő kissé szkeptikus bármiféle *Xbox* folt elfogadásával kapcsolatban. Az ok? Semmilyen *Linux* változat nem képes futni azon a rendszeren ha a felhasználó ki nem nyitja a dobozt és kicsit át nem alakít pár alkatrészt. Erre több lehetőség is van. *David Weinehall*, a 2.0-ás *Linux* rendszermag sorozat karbantartója az *Xboxot* beágyazott rendszernek látja és támogatja az olyan foltokat, amelyek a *Linuxot* jobb beágyazott operációs rendszerré teszik. A beágyazott fejlesztők számára az, hogy a *Linux* futtatásához át kell alakítani az alkatrészeket teljesen mellékes dolog. Az *Xbox* használók az *Xbox* támogatását a hivatalos rendszermag forrástól várják leginkább, hiszen az olyan terjesztők mint a Debian mindaddig vonakodnak támogatást nyújtani amíg a hivatalos rendszermag fa nem teszi ezt meg. Mindeközben azok a keményvonalas rendszermag-fejlesztők, akik úgy érzik a *Linuxnak* az összes elérhető rendszeren futnia kell, az *Xboxot* csak egy újabb kihasználható rendszernek látják, amely épp ezért igen kívánatos célterület. *Linus Torvalds* egy éve már elutasított egy *Xbox* foltot, de úgy hírlik már meggondolta magát. Igaz ugyan, hogy az *Xboxot* bizonyos *Digital Millennium Copyright Act (DMCA)* problémák övezik, a *Linux* változat fejlesztői megnézték a törvényt, és úgy gondolják minden amit csináltak megfelel a *DMCA* követelményeinek.

Az újítást és a baklövést csak egy hajszál választja el egymástól. Néhány fejlesztő úgy gondolja *Linus Torvalds* bizonyos hibákat követett el a jelenlegi rendszermag számozási sémában. A 2.6.8-as kiadást követően iszonyatos hibát találtak ami azonnali javítást követelt. Nem is a döntéssel volt a gond, hanem a programfejlesztés megszokott menetével, amit normál esetben egy hibajavító változat és a fejlesztés folytatása követett volna. Azonban a foltozást tartalmazó 2.6.9-es változat kiadása helyett *Linus* kiadott egy 2.6.8.1-es változatot. A negyedik verziószám hozzáadása teljesen ismeretlen volt a rendszermagfejlesztésben és rengeteg, a korábbi számozási elven alapuló eszközt összezavart. Később, amikor a 2.6.9 kijönni készült az egyik rendszermag kiadást 2.6.9-*final*-nak neveztek. A *final* szintén szokatlan volt és annyit jelentett volna, hogy ez a verzió lenne a hivatalos 2.6.9-es rendszermag.

Csakhogy, láss csodát, három nappal később egy másik, 2.6.9-nek nevezett kiadás is megjelent. Akkor mit jelent a *2.6.9-final* kiadásban a *final*? Ez megint csak összezavart sok parancsfájlt és *Russell King* felháborodásában megjegyezte: *„Én, a továbbiakban valahogy nem hiszek a fővonal semmilyen elnevezési elvében”*. *Matt Mackall* volt az első aki hangot is adott nemtetszésének *Linus* utóbbi időben tanúsított rendszermag verziószám-kezelése miatt, de hangjához hamarosan csatlakozott *Cliff White*, aki az egész *Open Source Development Labs* csoport nevében szólt, annak a csoportnak a nevében amely Linus-t a rendszermag fejlesztésére alkalmazta. Russell, *Geert Uytterhoeven*, *Christoph Hellwig* és *Martin J. Bligh*, valamennyien régi *Linux* fejlesztők, szintén felszólaltak a számozási anomáliák ellen.

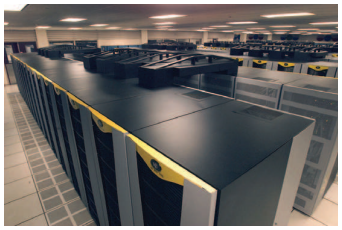
Zack Brown

Linux Journal 2005. február, 130. szám

Új termékek

Altix 3700 Bx2

A *Silicon Graphics (SGI)* nemrég bővítette választékát az *SGI Altix 3700* rendszer új változatával,



az *Altix 3700 Bx2*-vel. A *Bx2* akár *256 Itanium 2* processzort is képes befogadni, ide értve az új, 9 MB gyorsítótárral ellátott, 1,6 GHz órajelű Itanium 2 lapkákat is. A *Bx2* a *NUMAflex* globális megosztott memória alrendszer segítségével teszi elérhetővé a processzorok hatalmas teljesítményét az alkalmazások számára. Az új típus már az *SGI NUMAlink 4* összeköttetési megoldását használja, ezzel 6,4 GB/s-ra kétszerezve meg a szekrények közötti sávszélességet. Minden *Altix 3700 Bx2* gépbe 16 - 256 darab processzor, 8 GB - 24 TB globálisan címezhető memória és több mint 1500 *PCI-X* foglalat kerülhet. Folyamatos be/kiviteli sebessége meghaladja a 3 GB/s-ot.
www.sgi.com

Novell Linux Desktop 9

A *Novell* bejelentette a *Novell Linux Desktop 9*-et, mely egy a 2.6-os rendszermagra és

a *SuSE Linux* alapjaira épülő asztali operációs rendszer, és egyben irodai munkakörnyezet. Általános célú asztali gépekre is telepíthető, de az információs terminálok, a telefonos ügyfélközponti terminálok vagy a PC-t ritkán használó közönség igényeinek megfelelően is testreszabható. A *Novell Linux Desktop 9* tartalmazza az *OpenOffice.org Novell* által módosított, a *Microsoft* fájlformátumaival jobban együttműködni képes változatát, a *Novell Evolution 2*-t, a *Gaim* és a *Kopete* azonnali üzenetküldő programokat, a *Novell iFoldert*, a *Citrix ICA* ügyfelet és a *Firefoxot*. A *Linux Desktop 9* terjesztéssel végzett munkát a *Novell* a *YaST* és az *AutoYaSTUpdate Manager* felügyeleti eszközökkel, a *ZENworks Linux Management* csomaggal, vállalati szinten tesztelt foltokkal és frissítésekkel, *Linux* témájú képzésekkel és minősítésekkel, továbbá különféle szintű, világszerte elérhető támogatási szolgáltatásokkal igyekszik megkönnyíteni.
www.novell.com

Dell PowerEdge SC1425

A *Dell PowerEdge SC1425* 64 bites memóriacímzést, *DDR-2* memóriát és fejlett be/kiviteli

megoldásokat kínál egy egy egység magas kiszolgáló formájában. A gépet arra tervezték, hogy kiszolgálófürtök vagy webfarmok üzem közben cserélhető eleme legyen, ezért az *SC1425* a *Dell* nagyteljesítményű fürtcsomagjaiban érhető el, 8, 16, 32, 64, 128 vagy 256 csomópontból álló, a *Red Hat Enterprise Linux 3* 32 vagy 64 bites változatát futtató összeállítások részeként. További jellemzők: két darab *Intel Xeon EM64T* processzor *Hyper-Threading* támogatással és 800 MHz-es előoldali busszal, legfeljebb 12 GB *DDR-2* memória, beágyazott kettős *Gigabit Ethernet* hálózati csatoló és nagysebességű be/kiviteli képességek. A csomópont csomagok *Linux RHEL 3 AS/WS* alapúak, és négyféle összeköttetés használatára képesek: *Fast Ethernet*, *Gigabit Ethernet1*, *Myrinet* és *InfiniBand*.
www.dell.com

Qtopia 2.1

A *Trolltech, Inc.* a *Linux* alapú hordozható eszközökhöz készült *Qtopia 2.1* fejlesztői alaprend-

szert és felhasználói felületet két – *Qtopia Phone* és *Qtopia PDA* – változatban adta ki. A *Qtopia 2.1* újdonságai közé tartozik az érintőképernyős telefonok támogatása, a teljes képernyős kézírásos adatbevitel lehetősége, továbbá az új telefonos témák, amelyek révén kiterjesztők a felhasználók lehetőségei a linuxos hordozható eszközök fejlesztése terén. A *Qtopia 2.1 Phone Edition* továbbfejlesztett, immár az *MMS*-ek támogatására is képes üzenetküldő alkalmazást kapott, segítségével a *Qtopia Phone* használói képeket, szöveget és hangokat tartalmazó *MMS* üzeneteket hozhatnak létre és jeleníthetnek meg. A *Qtopia Flash* memória iránti igénye mérséklődött, ennek köszönhetően a 2.1-es változat már a korlátozottabb mennyiségű memóriával ellátott készülékeken is futtatható.
www.trolltech.com

Linux Journal 2005. 130. szám

Novell Linux Desktop: Az első benyomások

Amikor a Novell felvásárolta először a GNOME fejlesztői műhelyét, a Ximiant, majd az erős KDE-s kötődéséről ismert SuSE-t, mindannyian kíváncsian vártuk, vajon mi fog kisülni ebből a furcsa keresztezésből. Az eredmény a Novell Linux Desktop, amely egyesíti a SuSE kiváló hardverfelismerési képességeit és beállítási lehetőségeit a Ximian Evolution, az OpenOffice.org és a Mozilla Firefox által fémjelzett, letisztult munkakörnyezettel.

A ***KDE*** és a ***GNOME*** egyaránt elérhető maradt – mindegy is, hogy melyiket választjuk, a legjobb eszközöket kapjuk, nem muszáj mereven az egyik vagy a másik lehetőség mellett döntenünk. Csak kiragadott példa, hogy ***GNOME*** alatt a ***KDE***-s ***K3b*** az alapértelmezett CD-író program.

Bár a terjesztések egyre hasonlóbb alkalmazáskészleteket tartalmaznak, két terület még mindig maradt, ahol a ***SuSE*** előnye számottevő, ezek pedig a hardvertámogatás és a beállító eszközök. E területeken hosszú ideig a ***SuSE Professional*** volt a vezető, és tudását az ***NLD*** is örökölte. Egy ***Wacom Graphire*** tábla és egy kétmonitoros gép beállítása, üzembe helyezése alig néhány kattintás.

A ***SuSE*** terjesztéssel ellentétben a jelek szerint a digitális fényképezőgépek kezeléséhez nem települ semmilyen program, és a hálózati letöltések között sem találni ilyet. Az ***NLD*** a ***SuSE YaST*** és a ***Ximian Red Carpet*** megoldását egyaránt tartalmazza. Mindkettő telepítésre kerül, és mindkettőt használhatjuk a programok telepítésére és eltávolítására. A más környezetekről áttérő rendszergazdák számára ez minden bizonnyal zavaró. A linuxos „tegyük fel mindkettőt, majd a felhasználó választ közülük" szemlélet a való életben „gondoskodj róla, hogy minden rendszergazda egy kicsit másképpen telepítse, így minden személycsere végül valaminek a használhatatlanná válásával végződik" jelentést nyer.

Külső szerkesztőnk, ***Robert Love*** rajongói örömmel fogják látni, hogy új ***netapplet*** alkalmazása alapesetben is felkerül. Elég egy kattintás, és segítségével azonnal átválthatunk vezetékesről vezeték nélküli hálózatra, illetve váltogathatunk a különféle hozzáférési pontok között. Most már a legkisebb gondot sem okozhatja a vezeték nélküli hálózati hozzáférés beállítása, még a ***Mac OS X*** használói előtt sem kell szégyenkeznünk.

A ***Microsoft Windows*** alá készült, korábbról örökölt alkalmazások linuxos gépeken való támogatását kétféle módszerrel szokták megoldani: a megfelelő linuxos gépekre telepítenek valamilyen ***Windows***-emulációs programot, például a ***CodeWeavers CrossOver Office***-ét; vagy áthelyezik a windowsos alkalmazásokat egy valamilyen távoli-asztal-rendszert, például ***Citrix ICA***-t futtató gépre, a linuxos gépekre pedig az ügyfélprogramot telepítik.

Az ***NLD*** alapállapotában is képes a távoli asztalok kezelésére, köszönhetően annak, hogy az alaptelepítés részét képezi a ***gnomepro.com RDP***, ***VNC*** és ***ICA*** protokollt támogató ***Terminal Server Client*** alkalmazása. Ha az ***IBM 3270*** terminálokra íródott, jelenleg ***3270***-emulátorokon futó programok számából indulunk ki, akkor biztosak lehetünk abban, hogy sok felhasználó hosszú időn keresztül meg fogja még őrizni windowsos alkalmazásait. Mivel valószínűtlen, hogy az összes célprogramhoz elérhető lenne az emulátoros támogatás, ez a szolgáltatás kulcsszerepet játszhat az üzleti ***Linux*** tervezetek sikerében.

Volt olyan szolgáltatás is, amit nem sikerült kipróbálunk, és ez a Novell iFolder Linux Client, amely a felhasználói adatfájlokat egy kiszolgálóval szinkronizálja. Nyilvánvaló, hogy könnyebb az asztali gépek felügyeletét elvégezni, ha a felhasználói adatok egy fájlkiszolgálón találhatók, és a kiszolgálószobán kívüli merevlemezek száműzhetők. Ha a vállalatnál inkább hordozható gépeket használnak, akkor természetesen ez a régi stílusú megoldás nem működik, ilyenkor csak abban bízhatunk, hogy az ***iFolder*** gondoskodik a hordozható gépek használói által előállított adatok biztonsági másolatának elkészítéséről.

A ***Macromedia Flash*** beépülő modulja mellett felkerül a gépre az ***Adobe Acrobat Readere*** és a ***RealNetworks RealPlayere*** is, ingyenes lejátszóprogramként pedig a ***Totemet*** és a ***Rhythmboxot*** kapjuk.

A ***Novell Linux Desktop*** letisztult, rendkívül magas színvonalat képviselő munkakörnyezet kínál használóinak, aminek alsóbb rétegeiben az egyik legrugalmasabb ***Linux*** terjesztés végzi munkáját. Bár a telepítés, bevezetés terén még el kell végezni néhány kisebb fejlesztést, a munkaasztal készen áll a hétköznapi felhasználók kiszolgálására, és kiváló esélyekkel nyújtja be jelentkezését következő kedvenc ***Linuxunk*** szerepére.

Don Marti

A hónap szakmai tanácsai

Segfault memóriafoglaláskor

Lefordítottam egy alkalmazást, ám futtatáskor szegmentációs hibát kapok. A lefoglalt memória várt mérete nagy, 1000000-nál kevesebb. A hibát szerintem a következő sor okozza:

```
sample_space=calloc(sample_space_size,
➥sizeof(float));
```

A tényleges hívás valószínűleg a következő:

```
sample_space=calloc(820510, 4);
```

A program az alábbi paranccsal gond nélkül lefordítható:

```
gcc a2.c -o a2 -lm -lfftw3 -lz
```

Van valami ötletetek?
Mike, ➲ *m.giggey@utoronto.ca*

Memóriafoglalásnál ellenőrizned kell, hogy valóban megkaptad-e a kért memóriablokkot. Ha nem, akkor a memóriafoglaló függvények – például a `calloc` – `NULL` értékkel térnek vissza. A `NULL` által mutatott terület elérésére tett kísérlet szegmentációs hibát okoz. Lehet, hogy a felhasználói kvótát meríted ki. Az érvényben lévő korlátokat az `ulimit -a` paranccsal nézheted meg. Az értékek megváltoztatásával kapcsolatos tudnivalókat a `man ulimit` paranccsal jelenítheted meg, míg a `man calloc` segítségével a foglaló függvények működéséről tájékozódhatsz bővebben.
Chad Robinson, ➲ *chad@lucubration.com*

Javaslom, szerelkezz fel a ***gdb***-vel. A fordítást a `-g` kapcsolóval, a futtatást pedig a hibakeresőn belül végezd. Amikor fellép a szegmentációs hiba, végezhetsz egy visszakeresést. A ***gdb***-hez ***ddd*** névvel csinos grafikus felület is létezik. Érdemes megismerned mindkettőt, könnyebb lesz velük az életed.
Usman S. Ansari, ➲ *uansari@yahoo.com*

X Magic Cookie-k?

Nemrég frissítettem a 3.3-as ***KDE***-re, és gondjaim vannak a magic cookie-jaimmal, amikor egy-egy grafikus programot suval próbálok futtatni. Például, ha megpróbálom elindítani az xadminmenut, a következő jelzést kapom:

```
root@toad:/home/nathan/Desktop# xadminmenu
root@toad:/home/nathan/Desktop# Xlib: connection to
":0.0" refused by server
Xlib: Invalid MIT-MAGIC-COOKIE-1 key
Gtk-WARNING **: cannot open display: :0.0
```

Ha előbb átmásolom a ***.Xauthorityt*** a ***/home/nathan*** könyvtárból a ***/root/*** könyvtárba, akkor működik.
A ***magic cookie***-jaim köszönik, jól vannak. Próbáltam az `xauth add` paranccsal elérni, hogy a root magic cookie-ja egyezzen a sajátommal, ez a `toad:0` és a `unix:0` esetében mindig sikertelen, a `toad:10` és a `unix:10` esetében viszont sikeres. Valami teljesen kézenfekvő dolgot rontok el, mármint azon túl, hogy alaposan utána kellene olvasnom a témának? Vajon az az egyetlen megoldás, hogy készítek egy apró ***Perl*** parancsfájlt, amely bejelentkezéskor elvégzi a ***.Xauthority*** másolását, amit így nem kényszerülök kézzel letudni? Pillanatnyilag jobban örülnék, ha buherálgatás nélkül is működne a rendszer.
Nathan Oliphant, ➲ *nathan@oliphantparts.org*

A hibát több dolog is okozhatja:

1) Ellenőrizd, hogy a környezetedben be van-e állítva a `DISPLAY` változó, melynek értéke `:0.0`, `localhost:0.0` vagy valami ehhez hasonló legyen.
2) Ellenőrizd, hogy a fiókodnak van-e tulajdonlási joga ahhoz az ***X*** kiszolgálóhoz, amelyet használni próbálsz, vagyis a te fiókod-e az, amely jogot kapott a kijelző tulajdonlására/használatára. Azt, hogy mely ügyfélprogramok – ide értve a helyieket is – használhatják a kijelzőt, az `xhost` paranccsal derítheted ki. Kapcsolóiról a `man xhost` paranccsal tájékozódhatsz bővebben.
3) Az `Xlib: Invalid MIT-MAGIC-COOKIE-1 key` üzenet arra is utalhat, hogy `display 0` azonosítóval már fut egy ***X*** kiszolgáló a gépeden. Mielőtt elindítanád az xadminmenut vagy bármely más ***X*** alapú programot, add ki a `ps ax` parancsot, és ellenőrizd, hogy van-e már futó ***X*** folyamat. Ha igen, a ***Ctrl-Alt-Fx*** billentyűkombinációval próbálj meg átváltani erre a kijelzőre. (Az ***Fx*** a billentyűzet valamelyik funkciógombja, a 0-s kijelzőhöz általában az ***F7*** tartozik.)

A `man startx` parancs segítségével tekintsd át az ***X*** indításával kapcsolatos lehetőségeket. Az indítást rengeteg kapcsolóval és ezek számtalan kombinációjával lehet végezni.
Felipe Barousse Boué, ➲ *fbarousse@piensa.com*

Hiányzó könyvtár

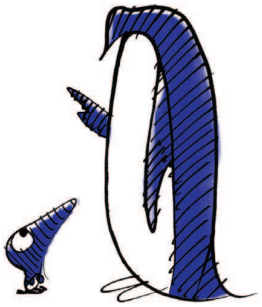
Hol tudnám beszerezni a ***libmp3lame.so.0*** könyvtárat? Az mplayerhez lenne szükségem rá. Amikor megpróbáltam telepíteni az mplayer ***RPM***-csomagját, a gép jelezte, hogy szükségem lesz erre a könyvtárra, bármi is legyen az.
David A. Barnett, ➲ *davecom@io.com*

Telepítsd a lame és a lame-libs csomagot. Red Hat alá, azt hiszem, a 3.92-2 a legújabb változat. RPM-csomagokat könnyedén tudsz keresni a www.rpmfind.net oldalon.
Chad Robinson, *chad@lucubration.com*

Microsoft hitelesítés linuxos VPN-en

Azt próbáltam kideríteni, hogyan kell beállítani egy linuxos ***VPN***-kiszolgálót, ha azt akarjuk, hogy a hitelesítést ***Microsoft Active Directory*** segítségével végezze. Nem szeretnék két külön listát vezetni a felhasználókról és a jelszavakról. Tudtok valamilyen forrást mondani, már amennyiben van ilyen, ahonnan tájékozódni tudnék?
Richard Rosenheim, ➲ *rlr0304@rlrmail.com*

Ugyanez a kérdés a Poptop SourceForge-os levelezőlistáján is felmerült már. A Poptop egy windowsos



A Linux Journal honlapján számtalan gond megoldásához találhattok további segítséget. A Sunsite tüköroldalait, a gyakori kérdéseket és az egyéb útmutatásokat a ➲www.linuxjournal.com honlapon olvashatjátok el. A rovatban közzétett válaszokat Linux-szakértők kis csapata készítette el. További kérdéseiteket szívesen fogadják (angol nyelven) a ➲www.linuxjournal.com/lj-issues/techsup.html címen, ahol csak egy kérdőívet kell kitöltenetek, de a bts@ssc.com címre levelet is írhattok. A levél tárgyában szerepeljen a „BTS” kulcsszó.

ügyfeleket is támogató linuxos VPN-kiszolgáló. Az egyik felhasználó több forrást is összegyűjtött, ezek alapján biztosan el tudsz indulni: ***sourceforge.net/mailarchive/forum.php?thread_id=5787492&forum_id=8250.***

Mielőtt belefognál, érdemes lenne feltenned egy egyszerű ***Samba*** fájlkiszolgálót a gépre, amely szintén a ***Microsoft*** címtárad szerint végezné a hitelesítést. Még ha később le is tiltod a szolgáltatásokat, mindehhez számos útmutató készült már, és megkapod azokat az alapszolgáltatásokat, amelyek a ***Poptop*** is igénybe tud majd venni.
Chad Robinson, ➲ *chad@lucubration.com*

A következő leírás valószínűleg segítségedre lesz: ***hawkerc.net/staff/abartlet/comp3700/final-report.pdf***

Az ***asia.cnet.com/enterprise/netadmin/0,39035505,39081966-39000223c-1,00.htm*** oldalt is érdemes lehet megnézned, bár itt régebbi változatokról van szó.
Felipe Barousse Boué, ➲ *fbarousse@piensa.com*

Illesztőprogramok hálózati kártyához

Nemrég új számítógépet kaptam, és úgy döntöttem, a régire ***Linuxot*** telepítek. Feltettem a 10.0-s ***Mandrake Linuxot***, és működik is minden, kivéve az internet-elérést. Semmilyen linuxos ***USB*** illesztőprogram nincs a modememhez, ezért egy elfekvő hálózati kártyát szereltem be a gépbe. Azt hiszem, ***Realtek 8139***-es, de ebben nem vagyok biztos. A hátán lévő matricán a ***Farallon PN993*** felirat látható, ám a ***Linux*** és a ***Windows*** egyaránt ***Realtek 8139***-nek ismeri fel.

Találtam egy ismertetőt a ***www.scyld.com/rtl8139.html*** címen, ami egész jó, részletes leírásnak tűnt. Letöltöttem a négy megadott fájlt, átmásoltam őket a linuxos gépemre, majd követtem az utasításokat. Amikor az első fordítási sorhoz értem…:

```
make KERNVER=`uname +-r` rtl8139.o
```

…lenyomtam az ***Entert***, amire válaszul hatalmas mennyiségű hibaüzenet szállt keresztül a terminálon. Más fordítási beállításokkal is próbálkoztam…:

```
gcc -DMODULE -Wall -Wstrict-prototypes -O6 -c
➥rtl8139.c
```

…de ugyanazokat a hibákat kaptam. Ekkor kipróbáltam a másik javasolt parancsot…:

```
gcc -DMODULE -D__KERNEL__ -O6 -c rtl8139.c
```

…és újfent ugyanazokat a hibákat kaptam. Megpróbálkoztam azzal is, hogy az egyes fordítási parancsokhoz a…:

```
-I/usr/src/linux/include -include /usr/src/
➥linux/include/linux/modversions.h
```

…sort adjam hozzá. Még csak ismerkedek a ***Linuxszal***, nem tudom, merre tovább. Megpróbálkoztam a ***Mandrake Linuxhoz*** mellékelt ***8139too*** illesztőprogram használatával is, de amikor kiválasztom, azt mondja ugyan, hogy települ, valójában viszont visszalép arra a képernyőre, amin megadom, hogy az illesztőprogramot kézzel szeretném kiválasztani. Végtelen ciklusba kerülök. Tudna valaki segíteni? Előre is köszönöm.
Sean Bowman, ➲ *poliwhirl74@mn.rr.com*

Sajnos a fenti címen található illesztőprogramot a 2.4-es rendszermagokhoz tervezték, a ***Mandrake 10.0*** viszont 2.6-os, pontosabban 2.6.3-as rendszermagot tartalmaz. A 2.6-os rendszermagok megjelenésével számos változás érintette az illesztőprogram-modulok fordításának módját, ezért sok még a 2.4-es rendszermagokhoz készült illesztőprogram nem használható tovább. Szerencsére a ***8139*** illesztőprogram régóta része a rendszermagfának, így szinte biztos, hogy ezeket a köröket nem kellene lefutnod.
A valódi válasz az, hogy ki kellene deríteni, a ***Mandrake*** miért juttat téged ilyen végtelen ciklusba az illesztőprogram telepítésekor. A legjobb valószínűleg az lenne, ha kérdésedet a ***Mandrake*** támogatási oldalán tennéd fel. Az oldal ingyenesen vehető igénybe, a legtöbb Mandrake-vonatkozású kérdésre gyors választ szoktak adni. (***www.mandrakeexpert.com***)
Chad Robinson, ➲ *chad@lucubration.com*

Mandrake Linux weboldala szerint a ***Realtek 8139*** kártyák támogatása megoldott; az ***RTL 8139*** támogatott hardvereszköz, az ***RTL-8139C*** és az ***RTL-8139D*** pedig hivatalosan tesztelt hálózati kártya. Ennek alapján azt javaslom, hogy a megfelelő illesztőprogramot betölthető rendszermagmodulként használd. Három választásod van, az első az `rtl8139`, a második a `8139too`, a harmadik pedig a `8139cp`. A legtöbb naprakész ***Linux*** terjesztés az utóbbi kettő valamelyikét tartalmazza. Rootként próbáld meg kiadni a `modprobe 8139too`, a `modprobe 8139cp` vagy a `modprobe rtl8139` parancsot. Valószínűleg a gépeden már a ***Mandrake 10*** alaptelepítés részeként, lefordítva és használatra készen várakozik a szükséges modul. Ha minden jól működik, akkor add hozzá az alábbi sort a ***/etc/modprobe.conf*** fájlhoz, így a modul betöltése önműködően, a rendszer indításakor megtörténik. Természetesen a működő modul nevét, például a ***8130too***-t kell megadnod:

```
alias eth0  8139too
```

Felipe Barousse Boué, ➲ *fbarousse@piensa.com*

Szinte minden esetben igaz, hogy ha egy hálózati kártyát támogat egy terjesztés, akkor semmit sem kell tenni ahhoz, hogy rendszerindításkor eleve működjön. Ha nincs fontos adatod a gépen, egyszerűen telepítsd újra, ekkor a ***Mandrake***-nek magától fel kell ismernie a kártyát, és ki kell választania a megfelelő illesztőprogramot.
Don Marti, ➲ *dmarti@ssc.com*

Linux Journal 2005. február, 130. szám

Ximba Radio: Grafikus felület fejlesztése az XM Satellite Radio programhoz GTK+/Glade segítségével

A Glade-ről azt mondják, hogy segítségével a grafikus programok prototípus-készítése egyszerű és gyors folyamattá tehető. Hogy a saját problémámnál, a számítógépemen lévő XM Satellite Radio programnál maradjak, úgy döntöttem, megvizsgálom, hogy pontosan mennyire egyszerű és gyors.

Ahogy az amerikai televíziózás egyre inkább elsüllyed a kitalált valóság feneketlen mélységeiben, egyre inkább azt veszem észre magamon, hogy kezdek visszatérni az elektronikus szórakoztatás gyökereihez, a rádiózáshoz. Ennek a médiumnak a legfrissebb megtestesülése a műholdas rádiózás, amely a csatornák széles választékának elérhetőségét biztosítja bárhonnan, ahova autóval el lehet jutni.
Mivel több időt töltök monitor előtt, mint az autóm kormánykerekénél, örömmel fedeztem fel a műholdas rádió egy számítógépes megoldását. Az ***XMPCR*** az ***XM Radio*** műholdas rendszer számára készült USB-csatlakozós vevőegysége, amelyet elsősorban Microsoft Windows rendszerekhez árusítanak. Linux alatt az eszközt az ***OpenXM*** projekt támogatja, ez egy Perl parancsfájlokból álló gyűjtemény, amely az eszköz vezérlését hálózati démonként működve oldja meg. Sajnos a démon egyetlen felhasználói felülete egy korlátozott szöveg alapú eszköz.

A projekt eredményeinek előzetes áttekintése

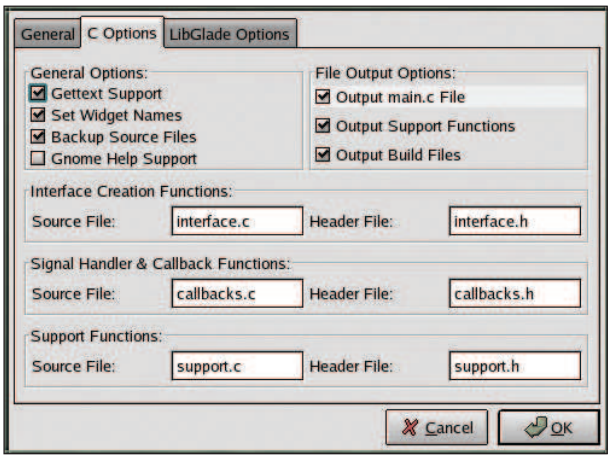
A ***Ximba Radio*** az ***OpenXM***-hez készült grafikus felületként született. A program egy végletesen egyszerű főablakot biztosít az aktuális csatorna információival, amely a csatornalistával és a felhasználó kedvenceivel bővíthető. Az ablak felső részén végighúzódó gombsáv biztosítja a rádió könnyű kezelését, az állomások közti mozgást, valamint a programbeállítási lehetőségek gyors elérését. A menüsor megadja a felhasználók számára egy hagyományos asztali program kényelmét.
A csatornalista többféle formátumban is megjeleníthető. A fő csatornalista ablaka minden csatornát megmutat, míg a kategóriák szerinti füleken csak a megfelelő csatornák szerepelnek. Külön füleken keresztül érhetőek el a felhasználó által kijelölt előadók és kedvenc csatornák, valamint a pillanatnyi folyamatelőzmény-lista. A kategóriafülek a ***Preferences (Beállítások)*** párbeszédablakban elrejthetőek, s ugyanitt tudjuk beállítani az előadásra és a kedvencekre vonatkozó megjegyzéseinket.

1. kép A főablak, a csatornalisták és a beállítások. A második változat az egyszerűsített formátumot mutatja a csatornalista elrejtésével.

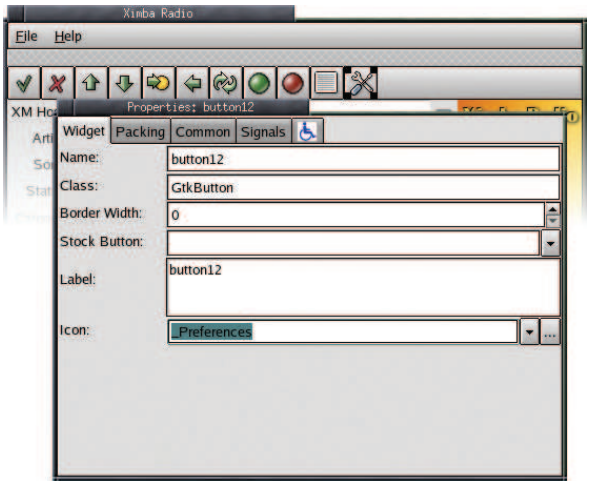
A ***Ximba Radio*** hátterét az ***OpenXM*** démon jelenti. Ez a ***Perl***-parancsfájl és a kapcsolódó ***Perl*** modul vezérli az USB kapun létrejövő kapcsolatot. A démon a beállításait a beállítófájlból vagy parancssori paramétereken keresztül kaphatja meg. A démonnal egy TCP-kapun keresztül tarthatjuk a kapcsolatot, amelyhez megadható az elfogadható ügyfelek listája. A démon futtatható Windows rendszerek alatt is.

Tervezési célok

Célom, hogy a ***Windows*** alatti változathoz hasonló képességekkel rendelkező, minél egyszerűbb felülettel bíró, könnyen beállítható programot hozzak létre. További feltétel volt, hogy a megvalósítás ne tartson tovább egy munkahétnél (40 óra).
Törekedtem arra is, hogy a felhasználói felület a lehető legnagyobb mértékben független maradjon a program törzskódjától. Egy program kódjának alkalmasnak kell lennie bármilyen megfelelő felhasználói felülettel való használatra, így jó tervezés esetén kevés munkával egy Webes felülettel való együttműködésre is könnyen rávé-



2. kép A Glade Options ablaka lehetővé teszi a kód előállítását, fájlnevek megadását és a nemzetközi szövegek támogatásának kiválasztását



3. kép A Glade toolbar-eszköze, és a tulajdonságok, amelyekkel ikonok és gombok hozhatók létre az eszközsávon belül

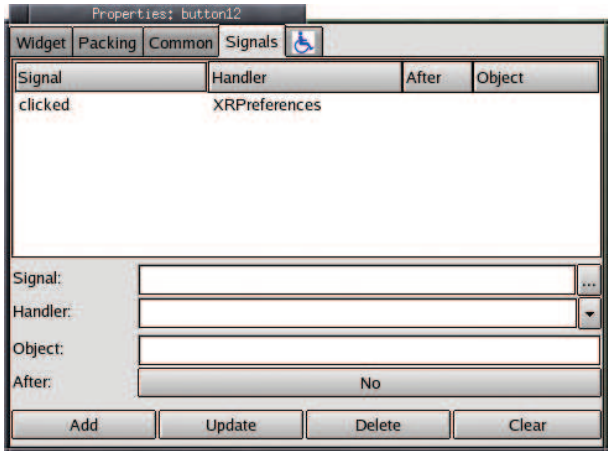
hetőnek kell lennie. Ez a cél összhangban áll a *GNOME* fejlesztői útmutatóival és a *Glade* jövőbeli terveivel egyaránt.
Ezeknek a céloknak megfelelően úgy terveztem, hogy egyetlen C fejállományt és egyetlen C modult fogok használni. A fejlesztés felgyorsítása és néhány időrabló probléma megoldása érdekében globális változók használata mellett döntöttem. Ne feledjük, hogy egy egyszerű asztali alkalmazásról és nem valami üzleti célú 24/7 hibatűréssel rendelkező programóriásról van szó. Ha jól kezeljük a kérdést, a globális változók használata a jövőbeli frissítéseknél kiküszöbölhető.

Ismerkedés a Glade-del

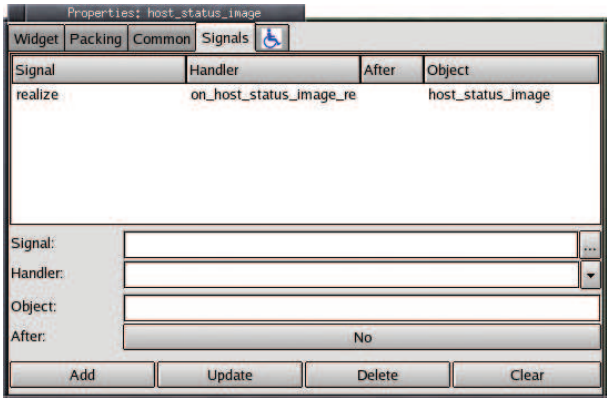
A célok meghatározása után belevágtam a *Glade*-del a felhasználói felület megformálásába – a következő részben részletezem ennek a folyamatnak a kóddal kapcsolatos vonatkozásait. Úgy állítottam be a *Glade*-et, hogy C-kódot állítson elő, az *Options* (lehetőségek) párbeszédablakban minden egyéb kérdésében elfogadtam az alapértelmezett beállítást. Itt a legfontosabbak a forrás- és fejállományok, amelyek a felhasználói felület meghatározásait tartalmazzák (*interface.c*), az eszköz-visszahívások (*callbacks.c*), és a *main.c* fájl előállításának beállításai.
A *Glade* a programok írására egy teljesen felépített környezetet hoz létre, amely tartalmazza a forráskönyvtárat (*src*) és a képfájlok tárolására szolgáló könyvtárat (*pixmaps*) is. A *GtkImage* eszközök által használt képeknek *xpm* formátumúnak kell lenniük. Az egyéb létrehozott fájlok között találjuk az *autoconf* és *automake* sablonokat és a *pot*-fájlokat a nemzetközi karakterláncok tárolására.
A nemzetközi támogatás nem kötelező, ezt a GNU *gettext* támogatása kezeli. Például az *interface.c* fájlban találhatunk *gettext* által engedélyezett karakterláncokat. Még ennek a lehetőségnek az engedélyezése esetén sem kell magunknak létrehozni a *pot*-fájlokat egyik nyelvhez sem, a Glade képes bármilyen szöveglánc használatára, amit alapértelmezett nyelvként adunk meg.
Tapasztalatom szerint a *Ximba Radio* prototípusának *Glade*-del történt elkészítése során a létrehozott fájlok közül mindössze kettő – a *main.c* és a *callbacks.c* – kézzel való módosítására volt szükség. Az előbbiben csak néhány kisebb kiegészítésre volt szükség a program beállításainak kezelésével kapcsolatos előzetesi választási lehetőségek terén. A *callbacks.c* fájl módosításainak zömét a C modulom, a *utils.c* felé történő hívásátadásai jelentették.
A projekt *Glade*-ben történő módosításai és a C-kód újbóli előállítása során a *callbacks.c* fájl új függvényekkel egészült ki. Szerencsére a *Glade* jól tudja, hogy mikor létezik már egy hívás, és egyszer sem írta felül a módosításaimat. Az viszont sajnos néha előfordult, hogy egy már létező függvényt újra hozzáfűzött. Időről-időre szükség volt ezeknek a függvényeknek a kézzel történő kitörlésére.
Ha *libGlade*-et használunk, amely a C-kód létrehozása helyett közvetlenül a Glade XML fájlt dolgozza fel, nem létezik ez a probléma, de a *libGlade* tárgyalása már meghaladja e cikk kereteit.

A felhasználói felület fejlesztése a Glade-del

A *Ximba Radio* két elsődleges ablakot igényelt, a *Főablakot* (*Main Window*) és a *Beállítások párbeszédablakot* (*Preferences Dialog*), valamint számos másodlagos előugró ablakot. A főablak gombsorát a *Glade* eszközsáv-eszközével hoztam létre, ehhez adtam kézzel a gombokat. A *GTK+* gombjai szöveget vagy képet tartalmazhatnak. A *Glade*-ben választhatunk alkalmazásképek (application images), beépített gombok (stock buttons) és beépített ikonok (stock icons) közül. A beépített gombok ugyanazokat az ikonokat használják, mint a beépített ikonok azzal, a különbséggel, hogy ezeknél az gyorsúgó nem elérhető. Emiatt én inkább az ikonok használatát javaslom, a stock button mezőt pedig hagyjuk üresen.
Az eszközsáv minden gombja egyetlen meghívható függvénnyel rendelkezik, amely a kattintáshoz (*click*) kapcsolódik. A hívandó függvénynek bármilyen nevet adhatunk, és igény szerint paraméterként magának az eszköznek a nevét is átadhatjuk. A kattintáshoz kapcsolódó függvények esetében ez utóbbi nem szükséges. A *realize* eseményekhez kapcsolódó függvényhívásokban – amiről hamarosan szó is esik – az eszköz nevét is átadjuk a függvénynek.
Három *GtkImage* eszközt helyeztem el a főablakon. Az első egy *State* (*Állapot*) ikon, amelyet a *Host name* (*Gépnév*) me-



4. kép Az eszközsávon lévő gombokhoz egyetlen függvényhívás rendelhető, amely a rá való kattintásra (clicked) lép működésbe

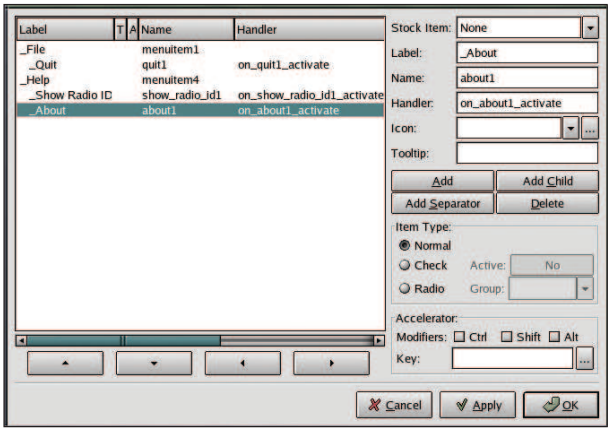


5. kép A State (állapot) ikonnak egy realize-függvényhívásra is szüksége van ahhoz, hogy megkapjuk az eszköz nevét és lehetővé váljon annak futási időben történő frissítése

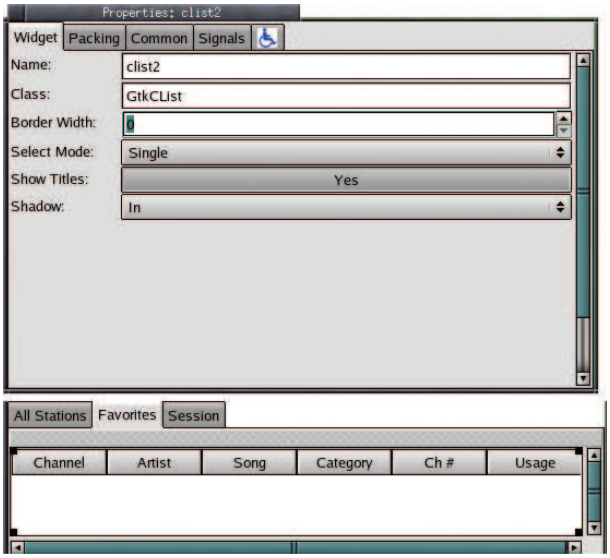
ző jobb felére raktam. A ***Remove*** (***Eltávolítás***) ikont állítottam be arra, hogy mutassa, ha nincs kapcsolat a démonnal, (a *Glade* egyébiránt, rengeteg beépített ikont tartalmaz).
A csatlakoztatott állapot feltüntetésére az ***Apply*** (***Alkalmaz***) ikont használtam. Az ikonok futási időben történő megváltoztatásához ennek a *GtkImage*-nek az eszközazonosítóját egy ***realize*** függvényhívásba mentettem. A használat közben ez az ikon az ***Off*** (***Kikapcsolt***) ikonra is beállítható, amely a csatlakoztatott de elnémított állapotot jelzi. A következő szakaszban meg fogom vizsgálni ezekhez a változtatásokhoz kötődő forráskódot is.
A ***Favourites*** (***Kedvencek***) gombok ábrája egy plusz jel. A *Glade* és a *GTK+* ezeket Add (hozzáadás) ikonoknak nevezi. Ezek a gombok egyetlen függvényhívással rendelkeznek, amelyet a hozzájuk kapcsolódó ***click*** (***kattintás***) esemény vált ki. A függvény az aktuális előadót vagy csatornát hozzáadja a megfelelő kedvencek listájához. A főablak tetején látható menüsort a *Glade* beépített ***Menüszerkesztőjével*** (***Menu Editor***) hoztam létre. A szerkesztőnek rengeteg beállítási lehetősége van, ezek közül én csak a ***Label*** (***Címke***), ***Name*** (***Név***) és ***Handler*** (***Kezelő***) tulajdonságokat használtam, ez utóbbit a menüpont kiválasztásakor hívandó függvény megadására.
Egy ***Notebook*** (***Jegyzetfüzet***) eszköz biztosítja a teljes csatornalista elérését csakúgy, mint a kategóriák, kedvencek és folyamatok szerinti listákét. Ezek kezelése a ***CList*** eszközön keresztül biztosított. A *Glade* teljes mértékben támogatja ezt az eszközt annak ellenére, hogy a *GTK+* az új kódokban már az újabb és bonyolultabb ***Tree and List*** (***Fa és Lista***) eszközt részesíti előnyben. Ezt a vitatható döntést kicsit később részletezni fogom majd.

A kódolás folyamata

A *Glade* a függvényhívásokhoz üres függvényeket hoz létre, amelyeket gyakran ***csonkoknak*** (***stub functions***) is neveznek. A csonkok lehetővé teszik, hogy a prototípus fejlesztése során egy egyszerű folyamat szerint haladjunk: megtervezzük a felhasználói felületet, létrehozzuk a kódot, megírjuk a függvényhívásokat, teszteljünk és ismételjünk. A hívások kódolásának nagy részét – a menüből való kilépés függvényeitől eltekintve – a felhasználói felület befejezése után végeztem el. Később tértem vissza a függvényhívások feltöltéséhez. Ez a módszer lehetővé tette, hogy a program elrendezésével kísérletezgessek anélkül, hogy túlságosan mélyen bele kelljen bonyolódnom az egyes részek működésébe. A módszer része annak a korábban említett törekvésnek is, hogy a felhasználói felület kódja minél inkább elkülönüljön a program forráskódjától. A két rész szétválasztásával lehetővé teszem a felhasználói felület jövőbeli módosítását anélkül, hogy ez komolyabban érintené az alapvető forráskódot. A felhasználói felület a függvényhívásokon keresztül kapcsolódik a programkódhoz, hiszen a felület eseményeit ezek képezik le a program szintjére, ahol ennek hatására végrehajtódik valamilyen tevékenység.
A függvényhívások különböző felületekkel rendelkeznek. A gombok ***click*** (***kattintás***) eseményei olyan hívásokat igényelnek, amelyek bemeneti paraméterként tartalmazzák a gomb-eszköz azonosítóját és a felhasználó adatait. A ***CList select-row*** (***sorkiválasztás***) eseményéhez tartozó hívás, amely egy sorra való kattintáskor következik be, öt paramétert vár. Ha hagyjuk, hogy a *Glade* hozza létre ezeket a függvényeket, hamar megismerhetjük ezeket a különböző felületeket. Valójában, mivel az API-hívások nem túl jól dokumentáltak – vagy legalábbis nem könnyű megtalálni a leírásokat – a parancsformátum megismerésének legjobb módja ha hagyjuk, hogy a *Glade* hozza létre a függvényeket.
A hívások kódjának kitöltését végezhetjük közvetlenül a ***callbacks.c*** állományban, de ezt a fájlt a jövőben, a ***lobGlade***-re való átálláskor már nem fogom használni, ezért ehelyett a paramétereket rendszerint egyenesen a utils.c fájlban lévő hasonló függvénynek adom át, amely a tényleges munkát végzi. Ezt az általános szabályt egy fontos kódrészlettel szegtem meg: az eszközazonosítók globális változókhoz való rendelését végző kód a callbacks.c fájlba került. Az 1. listán láthatjuk, hogyan használom egy hívásban a globális változót a ***Preferences*** (***beállítások***) párbeszédablak azonosítójának mentésére.
Több okból is szükségessé vált az egyes eszközök nyomonkövetése. Az első, hogy némelyik ikon a program állapotaitól függően dinamikusan változik. A második, hogy sok ablakot csak ideiglenesen kell megjeleníteni, ezek állandó



6. kép A Glade menüszerkesztője rengeteg lehetőséget kínál, de a prototípus elkészítéséhez ezek közül csak a Label, Name és Handler beállítására van szükség



7. kép A Glade teljes mértékben támogatja a CList eszközt, amely az egyszerű listák kezelésére alkalmasabb, mint a Tree and List eszköz

létrehozása és megsemmisítése felesleges lenne, sokkal kényelmesebb ezeket egyszer létrehozni és szükség szerint egyszerűen megjeleníteni vagy elrejteni. Végül pedig a *Glade* által létrehozott ***CList*** frissítésének futásidőben kell történnie. Az eszközazonosítókat tartalmazó változók érvényessége csak az ***interface.c*** fájlra terjed ki, ami azt jelenti, hogy azok a függvények, amelyek ezen a *Glade* által létrehozott fájlon kívül esnek, nem tudják egyszerűen módosítani ezeket az eszközöket.

A probléma megoldása végett minden olyan eszközhöz, amelyhez futásidőben is hozzá akartam férni, beállítottam egy felismerő eseményt. A *Glade* lehetővé teszi az ***interface.c*** fájlban definiált változók nevének megadását. Az említett eseményhez rendelt függvényhívás ennek a változónak az értékét kapja meg objektumparaméterként. A hívásban ez az érték egy olyan globális változóba kerül, amely az ***xr.h*** fájlban – a projekt számára általam létrehozott egyetlen fejállományban – került meghatározásra. Valamennyi globális változó érvényességét

1. lista A Preferences (beállítások) párbeszédablak az első kérés időpontjában jön létre, az eszközazonosító pedig globális változóba kerül

```
void
XRPreferences     (GtkButton       *button,
                   gpointer         user_data)
{

   /* Ha korábban még nem volt megnyitva,
    * hozzuk létre a párbeszédablakot.
    */

   if ( XR_Preferences_Window == NULL )
   {
      XR_Preferences_Window =
create_preferences();
      gtk_widget_realize(XR_Preferences_Window);
   }
   ...
}
```

2. lista A globális változók és függvények deklarációja az xr.h fájlban kap helyet

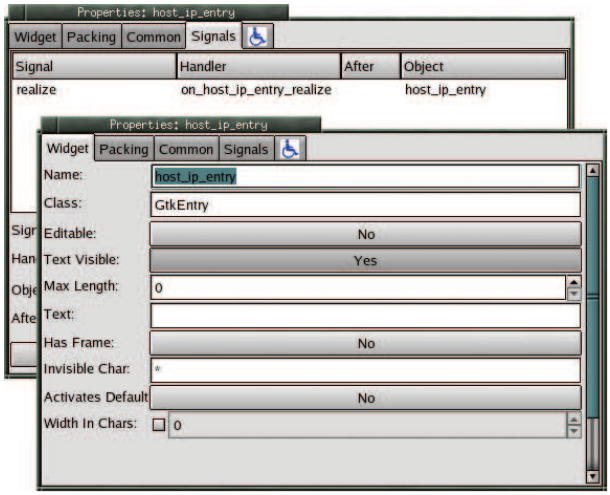
```
code:
#ifdef XR_CB_C
GtkWidget *XR_Msg_Window = NULL;
GtkWidget *XR_Msg = NULL;
void XRUMsg();
void XRUInit();
#else
extern GtkWidget *XR_Msg_Window;
extern GtkWidget *XR_Msg;
extern void XRUMsg();
extern void XRUInit();
#endif
```

3. lista A #define biztosítja a változók és függvények megfelelő elérhetőségét a C-modulokból

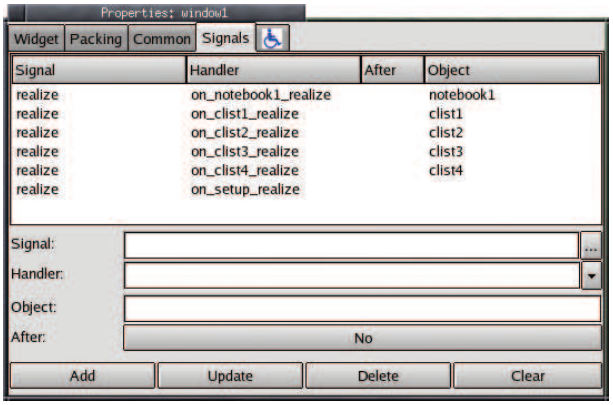
```
code:
#define XR_UTIL_C
#include <stdio.h>
#include <stdlib.h>
#include "xr.h"
...
```

a C-modul elején megadott ***#ifdef*** és ***#define*** előfeldolgozási direktívákkal szabályozzuk, amint az a 2. és 3. listákban is látható.

A módszer egyetlen problémája annak meghatározása, hogy mikor válik egy eszközazonosító elérhetővé. A ***realize*** eseményhez rendelt függvényhívás csak közvetlenül az



8. kép A realize események szerepe az eszközazonosítók egy függvényhívásnak való átadásában, amely azokat globális változókba menti



9. kép Közvetlenül a főablak megjelenítése előtt az eszközazonosítók és a CList ablakai többszörös függvényhívással globális változókba kerülnek

eszköz megjelenése előtt hajtódik végre, pedig néha már ez előtt is szükségünk van az eszközazonosítóra. Szerencsére létezik egy egyszerű megoldás. Az eszközt az ***interface.c*** állomány hozza létre még az eseménykezelők létrehozása előtt. Az eseménykezelő egy olyan függvény, amely a függvényhívást valamilyen eseménnyel kapcsolja össze. Ebből következően, mire a ***realize*** esemény beállítása megtörténik, már minden helyi változó érvényes értékkel rendelkezik. Ez lehetővé teszi azt, hogy egyetlen eszközhöz több olyan függvényhívást is létrehozzunk, amelyek mindegyike az adott eszköz ***realize*** eseményéhez kötődik és amelyek más eszközök eszközazonosítóit mentik el. Például a ***Ximba Radio main window (főablak)*** eszköze rendelkezik olyan ***realize***-hívásokkal, amelyek elmentik a ***Channel Listing*** (***Csatornalistázó***) ablak mind a négy előre meghatározott ***CList*** eszközének eszközazonosítóját. Erre szükség is van, hiszen kezdetben ezek a ***CList*** eszközök nem láthatóak, csak miután a főablak is azzá vált, viszont a frissítésüket azonnal meg kell kezdeni. Ha nem használnám a főablakot a ***CList*** eszközök azonosítóinak elmentésére, nem tudnám elkezdeni a csatornainformációkkal való frissítésüket sem mindaddig, amíg legalább egyszer meg nem jelentek a képernyőn.

Az eszközök dinamikus megváltoztatása szintén megkívánja az eszközazonosító mentését. Erre a ***Ximba Radio*** állapotát jelző ikon az egyik megfelelő példa. Az állapotjelző ikon megváltoztatásához szükség volt arra, hogy csak a ***GTK+*** ikonkészletéből származó ikont használjak, és elmentsem a ***Glade*** által létrehozott ***GtkImage*** eszköz azonosítóját. Amikor a felhasználó megváltoztatja a program állapotát – akár azzal, hogy megszünteti a kapcsolatot a démonnal, akár a némítás engedélyezésével vagy tiltásával – az állapotot jelző ikont egyszerűen egy ***GTK+*** függvényhívással megváltoztatjuk, amint az a 4. listán is látható. A ***GTK+*** beépített ikonjainak teljes készlete a hálózaton elérhető ***GTK+*** leírásában található meg.

Nem a globális változók kérdése a módszeremben az egyetlen kérdés, amelyet a tapasztaltabb fejlesztők vitathatnak. A másik a kifogásolt ***GTK+*** eszköz, a ***CList*** használata, amelyet oszlopos listának is hívnak. Az eszköz nem javasolt státusza azt jelenti, hogy ugyan része a jelenlegi csomagnak, de a jövőben már nem fejlesztik tovább és előfordulhat, hogy az ezután megjelenő ***GTK+*** csomagokból már hiányozni fog.

A ***CList*** eszköz helyettesítésére jelenleg a ***Tree and List*** eszköz használható. A ***Ximba Radio*** működése közben gyakran van szükség a listák elemeinek dinamikus bővítésére vagy szűkítésére. Ugyanakkor sem a ***CList***, sem a ***Tree and List*** eszköz nem volt igazán alkalmas ennek az egyszerű megvalósítására. Mégis mivel a ***CList*** tervezésekor a cél kifejezetten a listák és nem a kiterjeszthető fák kezelése volt, a kettő közül ezt találtam a kevésbé bonyolultnak. Az általános meghatározás szerint egy prototípusnak az a dolga, hogy általa az adott alkalmazásból gyorsan tudjunk előállítani egy olyan működőképes változatot, amely valamilyen szokványos felülettel és funkciókkal rendelkezik. A ***Glade CList***-támogatása ezt lehetővé teszi anélkül, hogy a ***Tree and List*** eszköz összetettségével bajlódni kellene. A választás igazi haszna persze majd akkor fog megmutatkozni, amikor a ***CList*** végső elrendezésével kell foglalkoznunk.

A ***Ximba Radio*** a listákat igen intenzíven használja. Minden csatornára, kategóriára és kedvencre vonatkozó információ oszloposan elrendezett listákban jelenik meg. Míg a teljes csatornalista és a kedvencek listája statikusak, melyek soha nem változnak meg teljes egészükben, a kategórialista dinamikus. A felhasználók a kijelzőn keresztül kapcsolhatják ki vagy be ezeket, megkönnyítve ezzel a saját beállításaiknak megfelelő állomások kikeresését. A kategórialista dinamikus mivoltának kezeléséhez jó hasznát vettem a ***GLib*** két irányban láncolt listájának, a ***GList***-nek. Bármennyire összetett is egy program, csak kevés esélye van rá, hogy nélkülözze a láncolt listák megjelenését, a ***GList*** használata szerencsére meglehetősen egyszerű. Létezik az egy irányban láncolt változat is, a ***GSList***, de nem sokkal bonyolultabb a kétirányú ***GList*** használata sem. Annak a lehetősége pedig, hogy a listában mindkét irányban szabadon mozoghassunk, megér annyi fáradtságot, amennyit a ***GList*** igényelhet.

Egy másik kelepcére a program tesztelésénél számíthatunk, amennyiben ***pixmap***-eket használunk. A ***Ximba Radio*** kü-

4. lista Ez a függvény változtatja meg az állapotikont a kapcsolat állapotának megfelelően, a GTK+ által Apply ikonnak nevezett ikon segítségével

```
code:
gtk_image_set_from_stock(GTK_IMAGE
                         ➥(XR_Status_Image),
   GTK_STOCK_APPLY,
   GTK_ICON_SIZE_BUTTON);
```

5. lista Ez a két függvény fájlba írja ki a beállítások adatait

```
code:
void
XRUSavePrefs()
{
   ...

   /* A beállítások kiírása */
   fprintf(fd, “hostname:%s\n”, prefs.hostname);
   if ( prefs.daemondir )
      fprintf(fd,
“daemondir:%s\n”,prefs.daemondir);
   else
      fprintf(fd, “daemondir:\n”);
   fprintf(fd, “favorites:%d\n”,
      (int)prefs.enable_favorites);
   fprintf(fd, “channels:%d\n”,
      (int)prefs.channel_windows);
   fprintf(fd, “performance:%d\n”,
      prefs.performance);

   /* A kategórialista futtatása és mentése
    * az állapotukkal együtt
    */

   g_list_foreach(prefs.categories,
      SavePrefsCategory, fd);

   ...
}

static void
SavePrefsCategory(
   CatEntryT    *catentry,
   FILE         *fd
)
{
   fprintf(fd, “category:%s:%d\n”, catentry-
>name,
      catentry->state);
}
```

lönböző helyeken tüntet fel emblémaként egy-egy ***pixmap***-et. Az alapértelmezett src könyvtárból nem találja meg a program a ***pixmap***-et anélkül, hogy előtte létre ne hoznánk egy teljes fordítást:

```
./configure –prefix=<telepítési könyvtár>
make
make install
```

A folyamat során a ***pixmap*** képek átmásolódnak az előtagként megadott telepítési könyvtár alatt lévő ***pixmap*** könyvtárba. Például, ha a telepítési könyvtár a ***/usr/local/ximbaradio***, a pixmap fájlok a ***/usr/local/ximbaradio/share/ximbaradio/pixmaps*** könyvtárba kerülnek. A fentebbi telepítést elvégezve, a lefordított program rendben megtalálja a ***pixmap***-eket. Amennyiben megváltoztatjuk a ***pixmap*** fájlokat, a `make install` lépést újra meg kell tennünk. Lefordított emblémákra válthatunk – azaz a bináris állomány részévé tehetjük azokat, azért, hogy a ***pixmap***-ek helyének változása ne lehessen befolyásoló tényező. Ez elérhető a ***support.c*** fájl módosításával. A korábbi programoknál ezt az eljárást követtem, de ezen technika ismertetése már meghaladná a cikk kereteit.

Végső elemzés

A teljes programot összesen körülbelül harminc órányi munkával sikerült létrehoznom. Ennek nagy részét a központi kód írásával töltöttem. A felhasználói felület kódja nagyjából tíz óra alatt készült el. A program minden fontosabb vonatkozásában megfelel a Windowsos változat felhasználói felületének, a beállítások is könnyen kezelhetőek. A felhasználói felület kódja független a program törzsének kódjától, bár a ***callbacks.c*** fájlon keresztül még jelen van némi függőség.

A ***Ximba Radio*** fejlesztése tovább fog folytatódni, a tervek közt szerepel hangszabályozási lehetőségek és egy ***GStreamer*** alapú reflektor hozzáadása. A ***reflector***, a ***Ximba Radio*** és az ***OpenXM*** egyesítése lehetővé fogja tenni a PC-alapú ***XM Radio*** műholdas szolgáltatás távoli elérését.

A ***Glade 3*** fejlesztés alatt áll, és számíthatunk arra, hogy a kódgeneráló részt majd eltávolítják. Mivel már elég régóta fejlesztik ezt a változatot, és a kibocsátása sem tűnik túl közelinek, az általunk használt program által létrehozott kód felhasználása a ***Glade***-del való prototípuskészítéshez a közeljövőben még biztosan életképes lehetőség marad. Vagyis a prototípusfejlesztés továbbra is egyszerű és gyors a ***GTK+*** és a ***Glade*** felhasználásával.

Linux Journal 2004. szeptember, 125. szám

Michael J. Hammel szoftvermérnök és író, a texasi Houstonban él Brinda nevű feleségével és Ryann nevű lányával. A fáradhatatlan futó és teniszjátékos, aki odáig van a kutyáiért, Reba-ért és Baley-ért, valamint a számítógépéért. Szabadidejét öregedő karfák ápolásával és szakadt kanapépárnák tisztításával tölti, és azt kérdezi magától, hogy miért nincs egy perc szabadideje sem.
Honlapja a ➲ graphics-muse.com címen érhető el, ő maga pedig a mjhammel@graphics-muse.org levélcímen.

Verziókezelés az Arch-al (2. rész)
Karbantartás és kifinomult használat

Helyezzünk üzembe hatékony verziókezelő rendszert kiszolgálónkon, mindössze a web és az SSH programok használatával. Bemutatjuk, mi szükséges egy program-projekt Arch alapú karbantartásához.

Az ***Arch***, az egyik mostanában megjelenő fiatal verziókezelő, fontos szerkezeti előnyt képes felmutatni az jó öreg ***Concurrent Version System (CVS)*** rendszerrel és társaival szemben. Decentralizált verziókezelő lévén, az ***Arch*** segítségével nagy mennyiségű fejlesztési munkát végezhetünk különleges előjogok igénylése nélkül. Az ***Arch*** ezen felül hatékony archívumközi műveleteket is nyújt, melyek jelentkezésre ösztönzik a harmadik felet.

A sorozat előző részében bemutattuk az alap ***Arch*** műveleteket: hogyan kérjünk ki kódot valamint miképpen lehet ágakat készíteni a távoli archívumokból. Most megtudhatjuk, hogyan vonhatjuk vissza a módosításainkat, hogyan adhatjuk ki saját archívumainkat nyilvános tükrökön keresztül és hogyan másoljuk változtatásunkat archívumból archívumba ha elfelejtettünk új ágat csinálni.

Az ***Arch*** program neve `tla`. Az `arch` programnév ugyanis már foglalt a ***POSIX*** szabványban, amely kiköti, hogy a ***/bin/arch*** parancsnak rendszerinformációkat kell visszaadnia. A `tla` súgójának futtatásával sok információhoz juthatunk. Amennyiben valamelyik parancs, például a `commit` paramétereire vagyunk kíváncsiak, érdemes lefuttatni a

```
tla commit -H
```

parancsot, hogy lássuk mit is tud a `tla commit` parancsa.

Változások mentése

Minden verziókezelő rendszer egyik nyilvánvaló előnye, hogy visszavonhatjuk egy vagy több változtatásunkat. Mindenki követ el hibákat, nem is ritkán, így aztán elég fontos hogy eszközeink rendelkezzenek valamilyen barátságos helyreállítási lehetőséggel.

A kikért fa helyi változtatásainkat nem tartalmazó változatához legkönnyebben a `tla undo` futtatásával térhetünk vissza. A parancs létrehoz egy ***undo-1/*** nevű könyvtárat, ahol az összes módosításunk található. Ha úgy tetszik, a `tla redo` parancs kiadásával egyszerűen kiadjuk újra érvényesíthetjük változtatásainkat. Például:

```
$ tla register-archive http://www.lnx-bbc.org/arch
$ tla get \lnx-bbc-devel@zork.net--gar/
  ➥lnx-bbc--stable bbc
$ cd bbc/
$ echo "BIG MISTAKE" > robots.txt
$ echo "#smaller change" >> Makefile
$ tla undo
$ tla redo
```

A `tla undo` parancs legnagyobb hasznát a „na-álljunk-csak-meg-egy-pillanatra" esetekben vehetjük, amikor a jelenleg folyó munkát kicsit félretennénk, valamilyen gyors változtatás érdekében. Az ***Arch*** belsőleg is használja az `undo` és `redo` parancsokat például az `update` vagy `star-merge` utasítások végrehajtásakor.

Egyetlen fájl visszaállítása

Amennyiben a hiba egyetlen állományra korlátozódik, nincs szükség a teljes változáshalmaz elmentésére. Az ***Arch*** egyetlen fájl visszaállításának problémáját úgy oldja meg, hogy az utolsó frissítés óta bekövetkezett változásokról egységes ***diff*** állományt készít. A ***diff*** aztán betölthető a visszaállítás módba kapcsolt ***patch*** programba, így a változtatások eltávolíthatók az állományból.

```
$ tla file-diffs robots.txt | patch -R
```

Amennyiben az állományt véletlenül letöröltük, a parancs végrehajtás előtt meg kell hívnunk a `touch robots.txt` parancsot. Állomány nélkül (még csak egy üres állományról van is szó), az ***Arch***-nak nincs mire támaszkodnia a ***file-diffs*** létrehozásánál. Ugyanakkor ha teljes változáskészletekkel dolgozunk az ***Arch*** sokkal intelligensebb.

Egész változtatáskészletek visszaállítása

Az ***Arch*** egyik nagy előnye elődjéhez a ***CVS***-hez képest, hogy támogatja a változtatás-készletek létrehozását és kezelését. A változtatás-készlet tulajdonképpen egyetlen

`tla commit` hívás során lejegyzett valamennyi szerkesztés, átnevezés, felvett és törölt állományok és naplóbejegyzések adatainak összessége.
Előfordul, hogy olyan változáskészletet küldünk el, amit nem kellett volna, esetleg valaminek csak az ideiglenes állapotát mentjük el, míg egy stabilabb változat betölti a helyét. Ezekben az esetekben visszavonhatjuk a változtatásokat ha fordított sorrendben játsszuk le őket:

```
$ tla replay --reverse \
  jrh@zork.net--projects/foo--bar--1.0--patch-4
$ tla sync-tree \
  jrh@zork.net--projects/foo--bar--1.0--patch-4
```

Az első parancs visszavonja az foo fa bar ágában található 1.0-ás verzió negyedik változtatáskészletét, még akkor is ha az nem a legfrissebb verzió. Mivel az adott változtatáskészlethez a naplóbejegyzéseket is mentjük, a `tla sync-tree command` parancs a `commit` naplót elvárásainknak megfelelően állítja vissza.
A ***patch-4*** változáskészlet továbbra is megtalálható a ***jrh@zork.net--projects*** archívumban, a fát tehát továbbra is ki lehet kérni e szerint az állapot szerint. A fenti parancsok kizárólag a jelenlegi munkaváltozatra voltak hatással. Amikor a fenti felhasználó lefuttatja a `tla commit` parancsot, új változáskészlet kerül fel, amely tartalmazza a ***patch-4***-et is.

Szemezgetés más ágakból

A `tla replay` parancs egyszerű „mégsem” funkciónál hasznosabb műveltekre is felhasználható. Az ***Arch*** egyik legvonzóbb tulajdonsága, hogy segítségével távoli források változáskészletei között szemezgethetünk anélkül, hogy a nekünk nem tetsző változásokat telepítenünk kellene. Vegyünk egy projektet, amelyet ***Bob*** tart karban. A projekt stabil ága (***foo--stable***) és kísérleti ága (***foo--experimental***) is ***Bob***-nál található. Az összes kiadást a legfrissebb ***branch--foo--stable--2.4.2*** stabil ágból készítjük. A kísérleti ágban tesszük közzé valamelyest hivatalosabb formában a kalandos próbálkozásokat.
Alice valamelyik kísérleti kódon szeretne dolgozni, ezért a munkához saját helyén kijelöli ***Bob*** kísérleti ágát:

```
$ tla my-id “Alice B. Hacker <abh@zork.net>”
$ tla make-archive -l abh@zork.net--work \
  sftp://abh@zork.net/home/abh/public_html/arch
$ tla archive-setup foo--hackery--0.0
$ tla register-archive
http://entar.net/~bob/fooarch
$ tla tag \
  bob@entar.net--code/foo--experimental--0.0 \
  abh@zork.net--work/foo--hackery--1.0
```

Miközben a kísérleti verzión dolgozik, ***Alice*** felfedez egy hibát ami fölött ***Bob*** tekintete úgy tűnik átsiklott. A javítás egyszerű, ezért félreteszi a jelenlegi munkáját a tla undo paranccsal és feltölti a javítást:

```
$ tla undo
$ vi buggy_file.c another_buggy_file.c
$ tla commit
M  buggy_file.c
M  another_buggy_file.c
* committed
  abh@zork.net--work/foo--hackery--1.0--patch-9
$ tla redo
```

Alice hamarosan befejezi a változtatásokat és jelzi ***Bobnak*** hol találhatóak az archívumai. ***Bob*** úgy dönt a változtatások elfogadhatóak a kísérleti kódágba és a `star-merge` funkcióval beolvasztja:

```
$ tla get bob@entar.net--code/
➥foo--experimental--0.0
$ cd foo--experimental--0.0/
$ tla register-archive http://zork.net/~abh/arch/
$ tla star-merge \
  abh@zork.net--work/foo--hackery--1.0
```

Miközben ***Alice*** változásnaplóját olvasgatja, ***Bob*** ráébred, hogy a kijavított hiba a stabil verzióban is létezik. Minthogy nem szeretné a teljes kísérleti kódot kiemelni a ***hackery*** ágból, ***Bob*** szemezgetve csak a hibajavítást végző változáskészletet tölti át:

```
$ tla get bob@entar.net--code/foo--stable--2.4.2
$ cd foo--stable--2.4.2/
$ tla replay \
  abh@zork.net--work/foo--hackery--1.0--patch-9
```

Változáskészletünk terjesztése

Alice és ***Bob*** együtt tudtak dolgozni, annak ellenére, hogy a két fejlesztő nem osztozott semmilyen, mindkettőjük számára elérhető közös rendszeren. Egyik fejlesztő sem készített dedikált kiszolgálót; egyszerűen a szokásos elterjedt protokollokat használták a ***http***-t, ***SSH***-t és az ***SFTP***-t. ***Alice*** archívumának előnye, hogy web könyvtárként elérhető az Interneten keresztül, akárcsak ***Bob*** hivatalos archívuma.
Az ***Arch*** segítségével ***Alice*** és ***Bob*** képesek voltak egymás egyedi archívumán és a különbségeken dolgozni, miközben semmi különlegeset nem használtak csak az ***Apache*** és az ***OpenSSH*** kiszolgálókat.

Aláírások

Ilyen sok kód küldözgetése az interneten keresztül mindig is zavarta kicsit a szabad program fejlesztőket, még ha csak tudat alatt is. A jelenlegi szakértői felülvizsgálati (peer review) rendszer gyorsan és hatékonyan megoldotta a rossz szándékú kódküldés problémáját, de még jobb lenne ha minden változáskészlet küldőjét kétséget kizáróan azonosítani lehetne.
A fejlesztők az ***Arch*** segítségével titkosítva aláírhatják a változtatáskészleteiket, így a saját megbízhatósági hálózatukon keresztül meg lehet állapítani a küldő kilétét. Habár ez nem feltétlenül bizonyíték a fejlesztő jó szándékára nézve, a hamisított küldeményeknek legalább gátat szab.
Ha az ***Arch***-ban titkosítást szeretnénk használni, először is készítenünk kell egy ***GnuPG*** kulcsot.

```
$ gpg --gen-key
```

Sajnos az aláírt archívumok némiképpen eltérő funkciókat kívánnak mint az aláírás nélküli változatok. Ezért külön archívumot kell fenntartanunk az aláírt küldeményeknek. A *tla make-archive* parancsot a `-s` kapcsolóval futtatva *GnuPG* aláírások tárolására is alkalmas archívumot kapunk:

```
$ tla make-archive -ls jrh@zork.net--signed \
  ~/SIGNED-ARCHIVE
$ tla my-default-archive jrh@zork.net--signed
```

Végül, néhány beállítás állományt kell létrehoznunk, hogy az *Arch* alá tudja írni a változtatás készleteket és ellenőrizhesse az aláírásokat. Először is a *tla* terjesztésében található *gpg-check.awk* nevű *awk* parancsfájlt kell feltelepítenünk valahová az *Arch*-ot futtató rendszeren. A *Debian tla* csomagok a */usr/bin/tla-gpg-check* helyre telepítik alapértelmezés szerint. Ha szeretnénk, hogy az *Arch* ellenőrizni is tudja az aláírásokat, a `~/.arch-params/signing/=default.check` állományba egyetlen sort kell felvinnünk:

```
$ mkdir ~/.arch-params/signing/
$ echo \
 ´tla-gpg-check gpg_command="gpg --verify-files
 ➥-"´\
  > ~/.arch-params/signing/\ =default.check
```

Amennyiben azt szeretnénk, hogy a kulcsok automatikusan töltődjenek le egy nyilvános kulcskiszolgálóról, akkor kiegészíthetjük a `gpg_command` parancsot például a `--keyserver pgp.mit.edu --keyserver-options auto-key-retrieve` kapcsolókkal. Mostantól az *Arch* minden `get` vagy `update` műveletek során szükség szerint letölti a *pgp.mit.edu* címről az archívum aláírásokat majd ellenőrzi őket.
Amennyiben szeretnénk ha az *Arch* automatikusan aláírná a változtatáskészleteket amelyeket -s kapcsolóval készített archívumokba küldünk, az `~/.arch-params/signing/=default` állománynak a következő egysorost kell tartalmaznia (a cím helyére a kulcs alkotásakor használt címet írjuk):

```
$ echo \
  ´gpg --default-key "<jrh@zork.net>"
  ➥--clearsign´ \
  > ~/.arch-params/signing/\ =default
```

Tükrök

A korábbi szemezgetős példánkban, *Alice B. Hacker* személyes archívumát weben tette elérhetővé. Ez ugyan kényelmes, de ha időnként lekapcsolódunk, problémákat vet fel. Mi van ha *Alice* a noteszgépéről szeretne dolgozni hosszú repülőútjai vagy vonatút közben? Nos, vagy változáskészlet tarlabdákat készít a `tla changes` segítségével, vagy kézzel, egyesével teszi fel noteszgépéről a különféle ágakat a `star-merge` segítségével amikor ismét netközelbe kerül. Szerencsére, az *Arch* lehetővé teszi olyan archívumok készítését is, melyek egyszerűen egy másik archívum tükörképei:

```
$ tla make-archive -ls --mirror-from \
  jrh@zork.net--signed \
  sftp://jrh@zork.net/public_html/arch/
```

A fenti `make-archive` parancsban *J. Random Hacker* egy internet-kiszolgálón hoz létre archívumot saját *public_html* könyvtárában. Miután a tükör archívum elkészült, `jrh@zork.net--signed-MIRROR` néven felkerül a `tla archives` listába. Mostantól egyetlen paranccsal adatot tölthetünk bele:

```
$ tla archive-mirror jrh@zork.net--signed
```

A feltöltő tükrökön kívül, melyek helyi archívum adatokat töltenek távoli rendszerekre, az *Arch* támogatja a lehúzó tükröket is melyekkel távoli források helyi tükörképeit készíthetjük el:

```
$ tla make-archive -ls --mirror \
  lnx-bbc-devel@zork.net--gar \
  /var/tmp/gar-cache
$ tla archive-mirror lnx-bbc-devel@zork.net--gar
```

Ez igen jól jöhet ha szétkapcsolódva dolgozunk, amikor a helyi ág már nem elegendő. A letöltő tükrök (pull mirrors) csak olvasási hozzáférést engedélyeznek a távoli archívum adataihoz amíg nincsenek a hálózatra csatlakozva.

Aláírt tükrök

A `jrh@zork.net--signed-MIRROR` archívum egyik hátránya, hogy maga is külön aláírt archívum. Így aztán *J. Random Hacker* kénytelen minden változáskészletet aláírni amikor az eredeti archívumból a tükörre másolja őket.
Bizonyos esetekben ez kívánatos is lehet. Például, jó, ha a kiadásfelelős külön-külön jóváhagy minden egyes változtatást ami kikerül a nyilvános kiszolgálóra. A legtöbb esetben azonban fontos lenne a változáskészlettel együtt egyszerűen csak átmásolni a már meglévő aláírásokat is. Ezt egy különleges állomány létrehozásával érhetjük el a `tla archive-mirror` parancsot futtató rendszeren:

```
$ echo jrh@zork.net--signed > \
  ~/.arch-params/signing/
  ➥jrh@zork.net--signed-MIRROR
```

Hálózaton kívüli munka (szórakozottaknak kis)

A tükrök rendkívül hasznos eszközök, de természetüknél fogva csak olvashatóak. Az egyetlen mód, amivel a változtatások eljuthatnak a tükörhöz az eredeti archívumon keresztül vezet a `tla archive-mirror` parancs használatával.
Vegyük *Alice* noteszgép tükör felállását. Mialatt az *Amtrak Coast Starlight* kocsijában utazik, előhúzza a noteszgépét és a `tla get` segítségével letölt valamennyi kódot az `abh@zork.net--work` nevű helyi tükörről. Valahol a *Willamette* völgyben ihletet kap és egy figyelemreméltó módosítást készít.
Ha megpróbálná elküldeni a változtatásait, egyfolytában a közvetlen tükörre írás hibaüzenetet kapná, ami azt jelenti, hogy a küldés sikertelen volt. Egyszerű megoldás lehetne,

hogy megvárja, míg elér egy internet elérési pontot, majd az undo és redo parancsokat használja:

```
$ tla undo ,changes-to-mirror
$ cd ~/real-project/
$ tla redo ~/mirror-checkout/,changes-to-mirror/
$ tla commit
```

Ez egészen jól működik, amennyiben a változtatásainknak egy változáskészletnél többre nincs szüksége. Azonban egy hosszabban elhúzódó fejlesztés során új helyi ágat szeretnénk létrehozni.
Miután megérkezett az ***Csendes óceáni*** partra, ***Alice*** felszáll a ***Chicago*** felé induló ***Zephyr***-re. Ez már hosszabb út, és azon kapja magát, hogy a ***bob@entar.net--code*** helyi tükrén található ***foo--stable--2.4.2*** kódon dolgozik. Néhány óra munka után úgy dönt, a változásokat új ágba veszi fel. Először is készít egy új archívumot és ágat a noteszgépén:

```
$ tla make-archive -l abh@zork.net--laptop ~/arch
$ tla my-default-archive abh@zork.net--laptop
$ tla archive-setup foo--laptop-hacks--1.0
```

Következő lépésként felteszi a tükör ágat az új archívumába. A `tla logs` parancsot parancsértelmező fordított aposztrófjai között futtatja így nem kell emlékeznie rá melyik foltozási szinten és verzión dolgozott éppen:

```
$ tla tag `tla logs -r -f | head -n 1` \
  foo--laptop-hacks--1.0
```

Végül, ***Alice*** kényszeríti a kikért másolatot, hogy azt higgye ez az első revízió az új ***laptop-hacks*** ágban:

```
$ tla sync-tree foo--laptop-hacks--1.0--base-0
$ tla set-tree-version foo--laptop-hacks--1.0
```

Ezzel a módszerrel a csak olvasható tükörről lekért másolatát a laptopján lévő írható olvasható archívumba telepítette át.

Ágkészítés Archívum nélkül

Tükör készítése egy hosszabb hálózaton kívüli időszakra ahhoz hasonlít, mint amikor útra csomagolunk: csak azt felejtjük otthon, amire gazán szükségünk lenne. Elég kiábrándító tud lenni, amikor az ember a notebookját hegyi lakának lámpa-dugójába bedugva ráébred, hogy a kimásolt változat bizonyos létfontosságú részei valamilyen ***HTTP*** archívumból származnának.
Szerencsére, ugyanezeket a technikákat felhasználhatjuk a kikért másolat új ágba mozgatásához akkor is, éppen ha nem tudjuk elérni a régi archívumot.
Alice közben egy ***Chicagoi Internet*** szalonban üldögélve lekérte a ***bar*** nevű projekt másolatát. ***Kaliforniába*** visszautazva, úgy dönt dolgozik egy kicsit a kódon. Újabb óra bámulatos erőfeszítés után ismét úgy találja, hogy ideje új ágat készíteni a munkájához.
Minthogy az eredeti forrás elérhetetlen, az ág felcímkézése lehetetlen. Szerencsére, a kikért fában a változásnapló és a történet információk nagy része megtalálható, így ***Alice*** ideiglenesen elmenti változtatásait a `tla undo` paranccsal, majd a kikért másolatra rákényszeríti saját új ágát:

```
$ tla archive-setup bar--train-ride--1.0
$ tla set-tree-version bar--train-ride--1.0
$ tla add-log-version bar--train-ride--1.0
$ tla import
```

Miután ezzel megvan, ***Alice*** lefuttatja a `tla redo` majd a `tla commit` utasítást. A ***Chicagoban*** leszedett változat neve most a `bar--train-ride--1.0--base-0`, módosított változata pedig a `bar--train-ride--1.0--patch-1` nevet viseli.
Bár a módszer nem tökéletes, a `star-merge` parancsot továbbra is gond nélkül használhatjuk az eredeti ágba és ágból való adatmozgatásra. Ha ***Alice*** úgy látja, hogy a bar projekt kellő mértékben érintett, valószínűleg összeolvasztaná a vezető archívummal és elkészítené a megfelelő ágat, amint ismét internetközelbe kerül.

Finomhangolásról legközelebb

Most már tudjuk, hogyan kell terjesztenünk az archívumainkat az Interneten és hogyan dolgozzunk távolról az ***Arch***-al. Még akkor is akad egy két trükk a tarsolyunkban, ha esetleg hibát követnénk el.
A sorozat következő, egyben utolsó részében bemutatjuk, hogyan tartsuk karban a központosított hivatalos archívumainkat az ***Arch*** megoszló munkarendszerének minden előnyét megtartva. Megtanulunk néhány archívumokkal kapcsolatos parancsfájlírási trükköt amelyekkel jelentéseket készíthetünk a fejlesztési aktivitásról, valamint létrehozunk egy tesztkörnyezetet.

Linux Journal 2004. december, 128. szám

- Mindenki követ el hibákat, nem is ritkán, így aztán elég fontos hogy eszközeink rendelkezzenek valamilyen barátságos helyreállítási lehetőséggel.
- Az Arch egyik nagy előnye elődjéhez a CVS-hez képest, hogy támogatja a változtatás-készletek létrehozását és kezelését.
- Az ***Arch*** segítségével ***Alice*** és ***Bob*** képesek voltak egymás egyedi archívumán és a különbségeken dolgozni, miközben semmi különlegeset nem használtak csak az ***Apache*** és az ***OpenSSH*** kiszolgálókat.

Nick Moffitt
San Francisco öbölvidékén élő hivatásos linuxos. Ő a GNU/Linux LNX-BBC Bootable Business Card változatának fejlesztőmérnöke és a GAR fordítórendszer szerzője. Amikor éppen nem programokat bütyköl, a tömegközlekedés történelmét tanulmányozza.

Központi hitelesítés Kerberos 5-tel (1. rész)

A Kerberos segítségével elfeledhetjük a fiókok felügyeletével kapcsolatos gondjainkat.

Elosztott ***UNIX/Linux*** alapú környezetben a fiókok kezelése rendkívül összetett és zűrös munka tud lenni, ha kézzel próbáljuk elvégezni. A nagyméretű rendszerekben különleges eszközökkel próbálnak úrrá lenni ezen a problémán. Írásomban ismertetem, hogy egy kisebb, akár egy három gépből álló hálózatban hogyan vehetjük hasznát ugyanezeknek az eszközöknek.

Az elosztott környezetek egyik gondja, hogy a jelszó- és árnyékfájlokat az összes gépen egyenként kell megváltoztatni, ha valamelyik fiók adatai módosulnak. A fiókok adatainak módosulása alatt a jelszóváltoztatásokat, a fiókok hozzáadását és törlését, a névváltozásokat (az ***UID***/***GID*** módosítások mindig sok bajjal járnak), az adott gépre szóló bejelentkezési jogok hozzáadását és törlését stb. értjük. Szó lesz arról is, hogy a ***Kerberos*** terjesztés révén hogyan válaszolhatjuk meg az elosztott számítási környezetekben felmerülő, hitelesítéssel kapcsolatos kérdéseket. A 2. részben a jogosultságkezeléssel kapcsolatos témákat tárgyaljuk.

A felhasználók számítógépekkel szembeni hitelesítését a legtöbb esetben jelszavakkal oldjuk meg, bár más módszerek – intelligens kártyák, biometrikus eljárások – is léteznek. A jelszavakat korábban a /etc/passwd fájl tárolta, az árnyékjelszavak megjelenése óta pedig a ***/etc/shadow*** fájlban találhatók. Mivel mindkét fájl helyileg tárolódik az egyes gépeken, naprakészen tartásuk sok gonddal jár. Erre a gondra kínálnak megoldást a címtárszolgáltatások, mint a ***NIS***, a ***NIS+*** és az ***LDAP***. Használatuk kapcsán ugyanakkor egy újabb probléma vetül fel: működésük a hálózatra alapul, és könnyen elérhetővé teszik a csak gyengén titkosított jelszavakat.

A ***Kerberos*** által megvalósított hitelesítő protokoll egyszerre biztosítja a hálózati szolgáltatások előnyeit, illetve teszi szükségtelenné a jelszavak számítógépek közötti továbbítását. Mindehhez két démont kell futtatni egy biztonságos kiszolgálón. A ***kulcselosztó központ (Key Distribution Center, KDC)*** démon a jelszóellenőrzési kérések kezeléséről és a ***Kerberos*** hitelesítő adatok, igazolványok előállításáról gondoskodik, ezeket ***jegyadó jegyeknek (Ticket Granting Ticket, TGT)*** nevezzük. A második démon, a ***Kerberos felügyelet (Kerberos Administration)*** segítségével távolról, a ***Kerberos*** démonokat futtató számítógépre való bejelentkezés nélkül is lehetővé válik a fiókok hozzáadása, törlése és módosítása. Ez kezeli a felhasználóktól befutó jelszóváltoztatási kéréseket is. A ***Kerberos*** használatakor jelszó hálózati továbbítására csak jelszóváltoztatáskor kerül sor, ekkor is erős titkosítás védelme alatt.

A ***Kerberos KDC*** egy ideiglenes igazolványt, egy ***TGT***-t ad a fióknak a felhasználó hitelesítésének folyamata során. Ezeknek az igazolványoknak jellemzően 10 vagy 24 óra az élettartamuk. Az élettartam változtatható, de ne legyen 24 óránál hosszabb. Ugyanis, ha ellopják a ***TGT***-t, a tolvaj az élettartam fennmaradó részében fel tudja használni. Az igazolványok érvényességének lejárása semmilyen gonddal nem jár, ha a ***Kerberost*** kizárólag hitelesítésre használjuk, ahogy írásomban ezt feltételezem. Ha viszont ***Kerberos*** alapú, „kerberizált” szolgáltatásokat használunk, akkor meg kell tanítanunk a felhasználóknak, hogy igazolványuk érvényességének lejártakor újat kell kérniük, függetlenül attól, hogy folyamatosan be voltak jelentkezve a rendszerbe.

A ***Kerberos*** alapjául szolgáló ötlet az ***MIT***-n született.

A ***Kerberos*** legújabb változata az 5-ös, leírása az ***RFC 1510*** dokumentumban található. Jelenleg két ***Kerberos***-megvalósítás érhető el szabadon. (Lásd az internetes forrásokat.) A ***MIT Kerberos 5*** a ***Red Hat Linux*** része, a ***Heimdal*** pedig a ***SuSE*** és a ***Debian Linuxban*** található meg.

A ***Kerberos 5*** megvalósításai a ***Microsoft Windowsba*** (***Windows 2000*** és újabb kiadások), a ***Sun Solarisába*** (***SEAM***, ***Solaris 2.6*** és újabbak) és az ***Apple Mac OS X*** operációs rendszerébe is bekerültek. Írásomban az ***MIT Kerberos*** terjesztésének használatát veszem alapul, ez ugyanis alapesetben is egyszerű jelszóminőség-ellenőrzést, jelszó-előregedést és jelszótörténetet biztosít.

Előfeltételek

A ***Kerberos*** alapú hitelesítésre két előfeltétel teljesítésével térhetünk át. Az első, hogy a ***Kerberos*** bevezetése által érintett gépek óráját kivétel nélkül a ***KDC***-t futtató gép órájához kell szinkronizálni. Ennek legegyszerűbb módja a ***Network Time Protocol (hálózati időprotokoll, NTP)*** telepítése az összes gépre.

A második feltételt már nehezebb teljesíteni. Minden fióknévnek, ***UID***-nek és ***GID***-nek azonosnak kell lennie az összes gépen. Erre azért van szükség, mert minden fiók új és független ***Kerberos***-fiókká változik, ezeket ***főfióknak (principal)*** nevezzük. Meg kell tehát vizsgálnunk az összes helyi ***/etc/passwd*** fájlt, és ellenőriznünk kell, hogy teljesül-e

ez a feltétel. Ha nem, akkor egységesítenünk kell a fiókokat. Ha ***Windows*** vagy ***Mac OS X*** alapú gépeket is be akarunk vonni, akkor ezeken a gépeken is át kell néznünk a fiókokat. Ha saját Linux-terjesztésünk ***Kerberosának*** használata mellett döntünk, akkor egyszerűen telepítsük a megfelelő csomagot. Ha viszont magunk akarjuk elvégezni a ***Kerberos*** lefordítását, akkor kövessük az alábbi útmutatót.

Az MIT Kerberos lefordítása és telepítése

1) Az internetes források között megadott URL-ek valamelyikéről töltsük le a forrást. Töltsük le a forráscsomag ***PGP***-aláírását is, és az alábbi paranccsal ellenőrizzük a forráscsomag sértetlenségét:

```
% gpg --verify krb5-1.3.4.targz.asc
```

2) Bontsuk ki a csomagot:

```
% tar zxvf krb5-1.3.4.tar.gz
```

3) Lépjünk át a forrás könyvtárába:

```
% cd krb5-1.3.4/src
```

4) Adjuk ki a következő parancsot:

```
% ./configure --help
```

Ezzel megtudhatjuk, hogy rendszerünkben milyen különleges beállításokat kell használnunk. Az alapértelmezett telepítési könyvtár a */usr/local/*. Ha más könyvtárba szeretnénk végrehajtani a telepítést, akkor a következő lépésnél használjuk a `--prefix=/új/könyvtár/elérési/útja` jelzőt.

5) A legtöbb esetben az alapértelmezett könyvtár tökéletesen megfelel:

```
% ./configure
```

6) Fordítsuk le a csomagot:

```
% make
```

Nálam volt valami gond a ***krb5-1.3.4/src/kadmin/testing/util*** könyvtár egyik fájljával, ezt azonban nyugodtan figyelmen kívül hagyhatjuk. Ha ilyen hibába ütköznénk, a `% make -i` paranccsal indítsuk újra a fordítást.

7) Ellenőrizzük, hogy minden rendben lezajlott-e:

```
% make check
```

8) Ha igen, telepítsük a csomagot:

```
% sudo make install
```

A fordítást soha ne végezzük rootként. A root jogosultságait csak akkor vegyük igénybe, amikor valóban szükséges, például a telepítési lépések során.
Ezzel telepítettük az ***MIT Krb5*** csomagját a */usr/local/* könyvtárba. Néhány további könyvtárat is létre kell hoznunk, illetve be kell állítanunk a rájuk vonatkozó engedélyeket:

```
% sudo mkdir -p /usr/local/var/krb5kdc
% sudo chown root /usr/local/var/krb5kdc
% sudo chmod 700 /usr/local/var/krb5kdc
```

Ha saját ***PAM***-modult szeretnénk fordítani, illetve feltétlenül szükségünk van rá, akkor a ***Red Hat*** által biztosított modult az alábbi lépésekkel bírhatjuk munkára. Töltsük le a forrást (lásd az internetes forrásokat), majd bontsuk ki:

```
% tar zxf pam_krb5-1.3-rc7.tar.gz
% cd pam_krb5-1.3-rc7
```

A `$PATH` környezeti változónak az általunk elsődlegesnek kiválasztott ***Kerberos***-terjesztés elérési útját kell tartalmaznia, ha valamiért több változat is telepítve lenne a gépen. Például:

```
% PATH=/usr/local/bin:$PATH
```

(Feltéve, hogy a telepítést a */usr/local* könyvtárba végeztük.) Ezután futtassuk az alábbi parancsot:

```
% ./configure
```

Záró lépésként az alábbi parancsokkal fordítsuk le és telepítsük a csomagot:

```
% make
% sudo make install
```

A tartomány létrehozása

A ***Kerberos tartomány (realm, birodalom)*** egy saját ***Kerberos*** adatbázissal rendelkező felügyeleti tartomány. Minden ***Kerberos*** tartomány saját ***Kerberos*** kiszolgálókkal rendelkezik. Tartományunk neve bármi lehet, ám tükröznie kell a DNS alapú világban elfoglalt helyünket. Ha az új ***Kerberos*** tartományt teljes ***pelda.com DNS***-tartományunk számára hozzuk létre, akkor ***Kerberos*** tartományunknak is azonos nevet adjunk: ***PELDA.COM***; csupa nagybetűvel, követve a kerberosos hagyományokat. Ha az új tartomány például a tervezési osztályt fogja lefedni, akkor válasszuk a ***TERVEZES.PELDA.COM*** nevet.
A tartomány létrehozásának első lépése a */etc/krb5.conf* fájl létrehozása, ez tartalmaz minden a tartománnyal kapcsolatos adatot. A ***krb5.conf*** fájlnak minden olyan számítógépen jelen kell lennie, amelyről el szeretnénk érni a ***Kerberos*** tartományt. A fájl tartalma a ***PELDA.COM*** tartomány esetében a következő lesz, feltételezve, hogy a KDC és a felügyeleti kiszolgáló egyaránt a ***kdc.pelda.com*** címen fut:

```
[libdefaults]
    # alapértelmezett tartománynév
    default_realm = PELDA.COM
[realms]
    PELDA.COM = {
        # kiszolgálók elérhetőségének megadása
        # valamint annak, hogy mely kapukon
        # fogadják a kéréseket
        # a szabványos kapuk a 88 és a 749
        kdc = kdc.pelda.com:88
```

```
        admin_server = kdc.pelda.com:749
    }
[domain_realm]
    # a DNS tartománynév megfeleltetése a Kerberos
    # tartománynévnek
    .pelda.com = PELDA.COM
[logging]
    # megadja, hogy az egyes szolgáltatásoknak
    # hova kell
    # írniuk naplózási adataikat
    kdc = SYSLOG:INFO:DAEMON
    admin_server = SYSLOG:INFO:DAEMON
    default = SYSLOG:INFO:DAEMON
```

A következő fájl, a */usr/local/var/krb5kdc/kdc.conf* a *KDC*-kiszolgáló beállításait tartalmazza. Ennek csak a *KDC*-démont futtató gépen kell jelen lennie. Mindegyik beállításnak ésszerű alapértéke van, ezért a legtöbb esetben egy üres fájl létrehozása is elegendő.

```
% sudo touch /usr/local/var/krb5kdc/kdc.conf
```

Az alábbi parancsokat a *KDC* szerepét játszó számítógépen kell kiadni. A

```
% sudo /usr/local/sbin/kdb5_util create -s
```

paranccsal létrehozzuk az új tartomány kezdeti *Kerberos* adatbázisát. A parancs bekéri az új tartomány adatbázisának fő jelszavát, majd a */usr/local/var/krb5kdc/.k5.PELDA.COM* fájlba írja. A parancs hatására a *Kerberos 5* fiókadatbázisban létrejön a főfiókok kezdő halmaza is. Ezeket a következő parancsokkal tudjuk kilistázni:

```
% sudo /usr/local/sbin/kadmin.local
```

A `kadmin.local:` parancssorba a `listprincs` parancsot kell beírnunk. A megjelenő lista a következő:

```
K/M@PELDA.COM
kadmin/admin@PELDA.COM
kadmin/changepw@PELDA.COM
kadmin/history@PELDA.COM
krbtgt/PELDA.COM@PELDA.COM
```

Pillanatnyilag még nem tudjuk használni a *kadmin* eszköz távoli változatát.
Mielőtt elkezdenénk létrehozni új tartományunk főfiókjait, meg kell adnunk a jelszavak kezelésére vonatkozó házirendet:

```
kadmin.local:  add_policy -maxlife 180days -minlife
?2days -minlength 8 -minclasses 3
?-history 10 default
```

A fentiekkel megadtuk az ezt követően létrejövő főfiókok mindegyikére érvényes alapértelmezett házirendet. A jelszavak maximális élettartama 180, minimális élettartama pedig 2 nap lett. Minden jelszónak legalább nyolc karakter hosszúnak kell lennie, és ezeknek a karaktereknek a következő öt osztály közül legalább háromból kell kikerülniük: kisbetűk, nagybetűk, számok, írásjelek, egyéb karakterek. A rendszer az utolsó tíz jelszót jegyzi fel, megelőzve az ismételt felhasználást. Ha a jelszavakat valamilyen szótár alapján is ellenőrizni szeretnénk, akkor egy `dict_file` értéket kell megadnunk:

```
[realms]
    PELDA.COM = {
        dict_file = /usr/share/dict/words
    }
```

A beállítás a *kdc.conf* fájlba kerül.
Készen állunk arra, hogy létrehozzuk saját felügyeleti főfiókunkat:

```
kadmin.local: addprinc janos/admin
```

A név természetesen saját nevünkkel egyezzen meg, a */admin* részt viszont hagyjuk változatlanul. Ezután kétszer be kell írnunk a főfiók jelszavát. Az új fiókot a következő paranccsal vizsgálhatjuk meg:

```
kadmin.local:  getprinc janos/admin
```

A kimenet az alábbihoz fog hasonlítani:

```
Principal: janos/admin@PELDA.COM
Expiration date: [never]
Last password change: Wed Dec 24 09:55:17 PST 2003
Password expiration date: Mon Jun 21 10:55:17 PDT
➥2004
Maximum ticket life: 1 day 00:00:00
Maximum renewable life: 0 days 00:00:00
Last modified: Wed Dec 24 09:55:17 PST 2003
➥(root/admin@PELDA.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 2
Key: vno 1, Triple DES cbc mode with HMAC/sha1,
➥no salt
Key: vno 1, DES cbc mode with CRC-32, no salt
Attributes:
Policy: default
```

A `quit` paranccsal lépjünk ki a *kadmin.local* programból, majd az alábbi utasítással indítsuk el a *KDC* démont:

```
% sudo /usr/local/sbin/krb5kdc
```

Az alábbi paranccsal kérjünk egy *Kerberos 5 TGT*-t:

```
% /usr/local/bin/kinit janos/admin@PELDA.COM
```

Majd vizsgáljuk meg a kapott TGT-t:

```
% /usr/local/bin/klist
Ticket cache: FILE:/tmp/krb5cc_5828
```

```
Default principal: janos/admin@PELDA.COM
Valid starting     Expires         Service principal
12/23/03 14:15:39  12/24/03 14:15:39
➥krbtgt/PELDA.COM@PELDA.COM
```

Gratulálok! Sikeresen túlestünk első ***Kerberos*** alapú hitelesítésünkön!
Most meg kell adnunk, hogy a felügyeleti fiók milyen jogosultságokat kapjon. Ezt a */usr/local/var/krb5kdc/kadm5.acl* fájlban található bejegyzések szabják meg. A ***janos/admin*** fióknak az alábbi sorral adhatunk felügyeleti jogot az összes főfiókra. A főfiókokat a * helyettesítő maszkkal jelezzük:

```
janos/admin@PELDA.COM  *
```

Mielőtt elkezdhetnénk hálózaton keresztül használni a felügyeleti démont (kadmind), készítenünk kell egy keytab fájlt, amely a tartomány létrehozásakor megadott kadmin főfiókok egyikének kulcsát tartalmazza:

```
kadmin.local:  ktadd -k /usr/local/var/krb5kdc/
?kadm5.keytab kadmin/changepw
```

Ezzel minden készen áll a ***Kerberos*** felügyeleti démon indításához. Indítása a következő paranccsal történik:

```
% sudo /usr/local/sbin/kadmind
```

A démon közreműködésével, a `kadmin` ügyfélprogrammal távolról is tudjuk kezelni ***Kerberos*** főfiókjainkat, nem kell bejelentkeznünk a ***KDC***-re. Ha ***Kerberos*** démonjainkat a rendszerindításkor önműködően szeretnénk indítani, akkor adjuk hozzá őket a ***KDC*** */etc/rc* fájljaihoz.
A korábban kapott Kerberos TGT-vel indítsuk el a távfelügyeleti programot:

```
% /usr/local/sbin/kadmin
Authenticating as principal janos/admin@PELDA.COM
with password.
Password for janos/admin@PELDA.COM:
```

Új fiókok hozzáadása

Az új fiókokat továbbra is hozzá kell adni a shadow fájlhoz vagy a jelszótérképhez. A titkosított jelszavak viszont nem ide kerülnek, ehelyett mindig egy új Kerberos főfiókot kell létrehoznunk, a jelszót pedig a KDC-be kell mentenünk.
A kadmin segédeszközzel:

```
% /usr/local/sbin/kadmin
```

egy normál felhasználó főfiókját a következő módon hozhatjuk létre:

```
kadmin:  addprinc janos
NOTICE: no policy specified for janos@PELDA.COM;
assigning "default"
Enter password for principal "janos@PELDA.COM":
Re-enter password for principal "janos@PELDA.COM":
Principal "janos@PELDA.COM" created.
```

A főfiók létrehozása során megadott jelszó lesz az, amelyet Jánosnak be kell írnia ahhoz, hogy egy ***Kerberos TGT***-hez jusson, vagy bejelentkezzen egy a ***Kerberos 5*** tartományba tartozó gépre.
Most az egyes fiókokhoz tartozó főfiókokat kézzel is létrehozhatjuk, de alkalmazhatjuk a lenti, áttérésről szóló részben ismertetett eljárást is.

Szolga KDC-k hozzáadása

Ha a ***Kerberost*** éles környezetben szeretnénk használni, akkor a rendszer hibatűrését további, szolgaként üzemelő ***KDC***-k hozzáadásával fokozhatjuk. Ehhez a mester ***KDC***-nek egy további, terjesztő szolgáltatásra van szüksége, amely mindig továbbítja a szolga kiszolgálók felé a ***KDC*** adatbázis frissített változatát. A szolga kiszolgálókra a terjesztő szolgáltatás fogadó oldalát kell telepíteni. A telepítéseket, beállításokat az MIT-s leírás tárgyalja bővebben.

Az ügyfelek beállítása

Egy számítógépet a legegyszerűbben egy ***csatlakoztatható hitelesítő modullal (pluggable authentication module, PAM)*** tehetünk képessé a ***Kerberos*** alapú hitelesítésre. Mivel ez ***Kerberos API*** hívásokat használ, működő /etc/krb5.conf fájlra van szüksége. Az első lépés tehát a /etc/krb5.conf fájl átmásolása a ***KDC***-ről az egyes ügyfélgépekre.
A ***Kerberost*** nemcsak felhasználók, de számítógépek hitelesítésére is használjuk, ezzel például megakadályozhatjuk, hogy hamis IP-című gépre jelentkezzünk be. Mindez csak akkor működik, ha mindegyik számítógép saját ***Kerberos*** főfiókkal rendelkezik, amelynek kulcsa, vagyis jelszava egy fájlba, pontosabban egy ***keytab*** fájlba kerül. A számítógépek főfiókjai különleges formátumúak:

```
host/<allomasnev>.pelda.com@PELDA.COM.
```

Első lépésként az összes ügyfélgéphez létre kell hoznunk egy-egy új főfiókot. Az alábbi parancsokban az `ugyfel1`-et használjuk számítógépnévként. Helyette értelemszerűen mindig az adott ügyfélgép állomásnevét kell alkalmazni. Lépjünk be az egyes ügyfélgépekre, majd adjuk ki az alábbi parancsokat:

```
% sudo /usr/local/sbin/kadmin
kadmin: addprinc -randkey host/
?ugyfel1.pelda.com@PELDA.COM
```

Ezzel az új főfiók véletlenszerű jelszót kap. Ezután írjuk ki a kulcsot egy ***keytab*** fájlba:

```
kadmin: ktadd host/ugyfel1.pelda.com@PELDA.COM
```

A parancs hatására létrejön a */etc/krb5.keytab* fájl. A */etc/* könyvtárhoz csak akkor kapunk írási engedélyt, ha a `kadmin` parancsot `sudo`-val futtatjuk. Ha csupán egy új főfiókot akarunk létrehozni, akkor nincs szükségünk kiemelt jogosultságokra. Ellenőrizzük a */etc/krb5.keytab* fájl tulajdonjogát és a rá vonatkozó engedélyeket. Írására csak a root kapjon jogot, ellenkező esetben sérül a gép biztonsága.
Többféle ***Kerberos 5 PAM***-modul is létezik, nevük egységesen ***pam_krb5***. Nagy részük az ***MIT Kerberos 5 1.3***-as válto-

zatának kiadásakor az ***API***-t érintően végrehajtott módosítások miatt ma már nem működik. A legjobban úgy járunk, ha a saját ***Linux*** terjesztésünkhöz tartozó ***PAM***-modult használjuk. A ***Kerberos 5 PAM***-modulok forrásból végzett fordításáról már ejtettünk szót.
Most adjuk hozzá az új ***PAM***-modult a rendszer hitelesítési készletéhez. Ezt – legalábbis ***Red Hat*** rendszereken – a */etc/pam.d/system-auth* fájl átírásával tehetjük meg. A fájlban szereplő bejegyzéseknek az alábbiakhoz kell hasonlítaniuk:

```
auth  required        /lib/security/$ISA/
➥pam_env.so
auth  sufficient    /lib/security/$ISA/pam_unix.so
➥likeauth nullok
auth  sufficient    /lib/security/$ISA/pam_krb5.so
➥use_first_pass
auth  required         /lib/security/$ISA/
➥pam_deny.so
account   required     /lib/security/$ISA/
➥pam_unix.so
account   [default=bad success=ok
➥user_unknown=ignore
?service_err=ignore system_err=ignore]
?/lib/security/$ISA/pam_krb5.so
password  required    /lib/security/$ISA/
➥pam_cracklib.so
?retry=3 type=
password  sufficient  /lib/security/$ISA/
➥pam_unix.so
?nullok use_authtok md5 shadow
password  sufficient  /lib/security/$ISA/
➥pam_krb5.so
?use_authtok
password  required    /lib/security/$ISA/
➥pam_deny.so
session   required    /lib/security/$ISA/
➥pam_limits.so
session   required    /lib/security/$ISA/
➥pam_unix.so
session   optional    /lib/security/$ISA/
➥pam_krb5.so
```

A módosítások elvégzése után minden olyan program, amelynek ***PAM*** beállító fájljában a `system-auth` ***PAM***-készlet szerepel – lásd a */etc/pam.d/* könyvtár egyéb fájljait – ***Kerberost*** fog használni a hitelesítési feladatokra.

Együttműködés nem Linux alapú ügyfelekkel

Ha már rendelkezünk működő ***Windows Active Directory (AD) KDC*** telepítéssel, akkor ezt is használhatjuk mester ***KDC***-ként a ***Linux***/***UNIX*** alapú gépek számára. Ebben az esetben a teljes kiszolgálótelepítést elhagyhatjuk, elég az ügyfelek telepítésére szorítkoznunk, illetve a */etc/krb5.conf* fájlban a ***Windows*** alapú ***KDC***-t kell megadni a ***UNIX*** alapú ***KDC*** helyett. A ***keytab*** fájlok létrehozásáról és másolásáról bővebben az internetes források tájékoztatnak.
Ha a csoportban windowsos gépek is vannak, természetesen ezekkel is csatlakozhatunk a ***UNIX*** alapú ***KDC***-hez, ám csak akkor, ha a gépek még nem tartoznak Kerberost használó windowsos ***AD*** tartományba, valamint a ***Kerberos*** és a ***Windows*** alatti fióknevek megegyeznek. Erről bővebben az internetes anyagokban lehet olvasni.
A ***Mac OS X*** alapú ügyfelek használata a ***Kerberos 5*** tartományban egyszerű, ahogy a ***UNIX*** alapú ***KDC***-k nevének megadása is a ***Mac*** gépeken. A fiókneveknek ebben az esetben is egyezniük kell.

Áttérés helyi jelszavakról vagy NIS/LDAP alapú rendszerről Kerberosra

Van egy működő ***Kerberos 5*** tartományunk és beállítottuk az ügyfeleket – a következő lépés a felhasználói fiókok áttelepítése. Eddig a fiókokhoz tartozó jelszavakat az egyes gépek a helyi */etc/shadow* fájlban vagy egy ***NIS***/***LDAP*** jelszótérképben tárolták. Ezek a jelszavak egyirányú kivonatoló algoritmussal vannak titkosítva, mely lehetetlenné, de legalábbis a gyakorlatban a szuperszámítógéppel nem rendelkezők számára nehezen kivitelezhetővé teszi a jelszavak feltörését, illetve a teljes átalakítást ***Kerberos 5*** formátumra.
A ***Kerberos*** alá történő áttelepítés elvégzésére kiváló megoldás a `pam_krb5_migrate` használata. (Lásd a forrásokat.) Ez egy vermelhető ***PAM***-modul, fel kell telepíteni néhány gépre, és minden alkalommal, amikor valaki bejelentkezik, a fiók aktuális jelszavával létrehoz egy új főfiókot a ***Kerberos 5 KDC***-ben.
Miután mindenki bejelentkezett ezekre a gépekre, az összes felhasználónak lesz egy ***Kerberos 5*** főfiókja. Ekkor a helyi fájlokban vagy a ***NIS***/***LDAP*** jelszótérképben található jelszavakat lecserélhetjük egy helyőrzőre, mint például krb5. Ettől a pillanattól kezdve a ***Kerberos PAM***-modul végzi a felhasználók hitelesítését. Az áttérést segítő gépekről eltávolíthatjuk a `pam_krb5_migrate` modult.

Kerberizált alkalmazások

Sikeresen életre hívtuk a ***Kerberost***, tehát elkezdhetünk rá támaszkodó szolgáltatásokat használni. Telepíthetnénk például ***Kerberos*** alapú `telnet`-et és ***FTP***-t, de ezek helyett használjunk inkább ***SSH***-t. Kerberizálhatjuk ***Apache*** webkiszolgálónkat és Mozilla böngészőnket is. A ***Kerberos*** alkalmazása előtt ezen szolgáltatások igénybe vételekor mindig jelszót kellett megadnunk, most azonban lehetővé vált, hogy mindezek az alkalmazások az elmentett ***Kerberos*** igazolványokat használják fel a megfelelő szolgáltatásoknál végzett hitelesítésre. Ezt a megoldást szokták egyszeri bejelentkezésnek nevezni (single-sign-on).

Linux Journal 2005. február, 130. szám

A cikkhez tartozó források elérhetősége:
➲ ***www.linuxjournal.com/article/7706***

Dr. Alf Wachsmann 1999 óta a Stanford Linear Accelerator Center (SLAC) munkatársa. Ő felelős az önműködő Linux telepítések minden mozzanatáért, egyaránt ide értve a farmok csomópontjainak, a kiszolgálóknak és az asztali gépeknek a kezelését. Munkája során elsősorban az aktív fájlkészletek (AFS) támogatásával, a Kerberos 5-re való áttéréssel, egy felhasználónyilvántartó tervezettel és felhasználói tanácsadással foglalkozik.

Fájlrendszerek indexelése a libferris segítségével

A teljes értékű szöveges és metaadat alapú keresés többé nem csupán álom. A libferris könyvtárral, mely a fájlok keresését tartalom és számos egyéb jellemző szerint teszi lehetővé, már ma is bárki számára elérhető.

A *libferris Project* 2001 elején indult, célja egy megosztott könyvtárként működő virtuális fájlrendszer létrehozása volt. A *libferris* egyetlen fájlrendszer felületen keresztül több fa jellegű szerkezethez biztosít hozzáférést. Mivel a *libferris* a rendszermag tér helyett a felhasználói térben fut, nagyszámú fa jellegű szerkezet elérhetővé tételére képes. Ezeket a forrásokat a *Linux* rendszermagból nehéz lenne elérni. A *libferris* használatakor minden fájlrendszer a `root://` *URI*-n keresztül érhető el, amely felöleli a rendszermag `file://` *URL*-jeit; a relációs adatbázisokat; az *XML* fájlokat és adatbázisokat; a hálózati, például *HTTP*/*FTP* alapú kiszolgálókat; az egyéb összetett fájlokat – *db4*, .tar és *RDF* –; továbbá a szabványos fájlrendszereket, mint az *ext3* és az *XFS*.

A fájl és a könyvtár fogalma *libferris* alatt egyetlen elvont fogalommá egyesül. Ezzel lehetővé válik, hogy a *libferris* fájlrendszerként fűzzön be például .tar állományokat. A .tar állomány például fájl és könyvtár is egyszerre.

A ***kiterjesztett jellemző (extended attribute, EA)*** felület különböző forrásokból származó adatokat tesz elérhetővé, ide értve a rendszermag *listxattr(2)* felületét, az *RDF*/*bdb* gyűjteményeket és a dinamikusan kibontott értékeket is. Dinamikus *EA* például egy kép szélessége. A képszélesség *EA* olvasásakor a *libferris* egy beépülő modul közreműködésével határozza meg az adott képfájlban lévő kép szélességét. Egy másik példa a hangfájlok mintavételezési gyakorisága. További *EA*-kat és ezek leírását a *libferris* webhelyén lehet találni. (Lásd az internetes forrásokat.)

A *libferris* kétféle típusú indexet képes létrehozni és lekérdezni, teljes szövegeset és *EA* típusút. A teljes szöveges indexek segítségével a fájlok között tartalmuk alapján tudunk keresni, míg az *EA* indexek a metaadatok alapján végzett kutakodást teszik lehetővé. A teljes szöveges vagy *EA* típusú lekérdezések feldolgozásához szükséges indexelő szerkezetek jelentős mértékben eltérők. A teljes szöveges indexek például az egyes szavakat tartalmazható dokumentumok listáját (fordított fájl) tárolva tudják feldolgozni az olyan kéréseket, mint a „keress meg minden a *libferris* szót tartalmazó dokumentumot”. Az *EA* indexeknek ezzel szemben a tartományokra vonatkozó lekérdezéseket is képeseknek kell lenniük kezelni, mint például „keresd meg a múlt hónapban módosított fájlokat”.

Miért épp a libferris?

Néhány érv amellett, miért érdemes a *libferrist* választani a fájlrendszerek indexelésére és a lekérdezések elvégzésére:

- Beépülő modulok segítségével, teljes szövegindexelés céljából képes a fájlokba ágyazott szövegek kibontására.
- Egyesített felületet ad minden az indexelés bemeneteként szolgáló forráshoz. Például a következő források egységesen indexelhetők a *libferrisszel*: *SleepyCat dbxml* fájlokban található szövegek, .tar állományok tartalma, *mbox* fájlokban vagy relációs adatbázisok által tárolt anyagok.
- A fájlokban található metaadatokat szintén lehet indexelni és keresni. Például a zenei fájlokban található *ID3* címkék alapján előadó szerinti keresésre nyílik lehetőség.
- Az alapszintű hozzáadó/lekérdező parancsok minden beépülő modulnál azonosak, az indexelő megoldások között tehát könnyedén lehet váltani.
- Kombinációs keresések szövegekre és a fájlrendszer kiterjesztett tulajdonságaira. A *ferris-search* keresőeszköz segítségével több keresést is egyetlen eredménykészletbe lehet egyesíteni.
- Fájlok keresése olyan metaadatok alapján, amelyekkel korábban rendelkeztek.
- Fájlkeresés *felügyelt gépi tanulás (Supervised Machine Learning, SML)* alapú szűréssel – gyakorlatilag szemétszűrés a fájlrendszeren belül. Az *SML* ismertetése sajnos túlmutatna írásom keretein.

A *libferris* beépülő modulokkal kezeli mindezeknek az indexeknek a megvalósításait. A teljes szöveges indexek esetében a következő megvalósítások bármelyikét, sőt, akár mindegyikét is használhatjuk: belső, fordított fájlokra épülő formátum; *gcj*-vel lefordított *Apache Lucene*, relációs adatbázis hátterű *ODBC*; *Xapian* vagy *TSearch2* modul *PostgreSQL*-lel. Az *EA* indexek esetében ugyancsak belső, fordított fájlokra alapuló megoldást; *LDAP*-ot; *gcj*-vel fordí-

tott *Apache Lucene*-t; relációs adatbázis hátterű *ODBC*-t vagy natív, némi *PGSQL*-lel kiegészített *PostgreSQL*-t választhatunk. Általános használatra a teljes szöveges indexekhez a *Xapian* vagy a *TSearch2* az ajánlott, *EA* indexekhez pedig a *PostgreSQL* vagy az *ODBC*.
A *PostgreSQL* modulok hasonlóak az *ODBC* alapúakhoz, ám ezek *PGSQL*-t és egyéb, kifejezetten a *PostgreSQL*-re jellemző szolgáltatásokat használnak. Ha a teljes szöveges indexeléshez a *PostgreSQL TSearch2* beépülő modulját használjuk, akkor *PostgreSQL* kiszolgálónkon egy sablon adatbázist kell létrehoznunk. A részletekről bővebben az internetes források közt lehet olvasni.
A *libferris* minden indexet a saját könyvtárába helyez. Az alapértelmezett teljes szöveges és az *EA* index rendre a *~/.ferris* könyvtár *full-text-index* és *ea-index* könyvtárába kerül. Az indexek létrehozása a *ferriscreate* csomag `fcreate` vagy `gfcreate` parancsával történik. A *libferris* más eszkö-

1. ábra Teljes szöveges Xapian index létrehozása a gfcreate segítségével

zeihez hasonlóan itt is igaz, hogy a *gf* előtaggal ellátott program ugyanazt teszi, mint az *f* előtagú, ám előbbi *GTK+2* felülettel is rendelkezik. A továbbiakban mindkét típusú index létrehozásáról, feltöltéséről és lekérdezéséről lesz szó.

Teljes szöveges indexelés

Vágjunk a közepébe: először létrehozunk egy teljes szöveges indexet a */tmp* könyvtárban, hozzáadunk néhány fájlt, majd az index használatával elvégzünk egy lekérdezést. Hozzuk létre az új index könyvtárát, majd a `gfcreate` segítségével magát az indexet is:

```
$ mkdir /tmp/text-index
$ gfcreate /tmp/text-index
```

A `gfcreate` grafikus felületének bal oldali lapján jelennek meg a fő *MIME* típusok, a *misc (speciális)* lapon pedig a létrehozható, de a *MIME* típusoktól különböző dolgokat találjuk. A *misc* kiválasztása után egy második lapszinten az összes elérhető indexformátum megjelenik. Az *1. ábrán* követhető, hogy *Xapian* teljes szöveges indexet választottam, a szótőképzéshez angol nyelvet jelöltem meg, illetve nem akarom megkülönböztetni a kis- és nagybetűket.

Amikor hozzáadjuk a fájlokat a teljes szöveges indexhez, a *libferris* az *as-text EA* használatával próbálja előállítani a fájl szöveges formáját. Az *as-text EA* támogatására számos beépülő modul készült; a *PDF* és a *HTML* fájlok, a *man* oldalak és a *djvu* képek egyaránt támogatják az *as-textet*. A `findexadd` és a `findexquery` eszközöknek a `-P` parancssori kapcsolóval adhatjuk meg, hogy melyik indexet használják. Az alábbi példában a *Samba 3.0.3* csomag egy *man* oldalát és egy *PDF* fájlját használjuk bemenetként. Mivel a tényleges elérési útvonalak az egyes *Linux* terjesztések esetében eltérők lehetnek, a fájlok előtti előtagokat a `/.../` karakterlánccal helyettesítettem:

```
$ findexadd -P /tmp/text-index \
/.../samba-3.0.3/docs/Samba-HOWTO-Collection.pdf
$ findexquery -P /tmp/text-index samba
ID 1 99% [file://.../Samba-HOWTO-Collection.pdf]
Found 1 matches at the following locations:
file://.../Samba-HOWTO-Collection.pdf
$ findexadd -P /tmp/text-index /.../samba.7.gz
$ findexquery -P /tmp/text-index smbstatus
ID 1 100% [file://.../Samba-HOWTO-Collection.pdf]
ID 5 93% [file://.../samba.7.gz]
Found 2 matches at the following locations:
file://.../Samba-HOWTO-Collection.pdf
file://.../samba.7.gz
```

A `findexquery` legfontosabb kapcsolói a `-P`, amely az indexfájl elérési útját adja meg; a `--ranked`, amely rangsorolt teljes szöveges lekérdezések indítására utasít; valamint a `--xapian`; amellyel nyers *Xapian* formátumú lekérdezéseket adhatunk át a háttérszolgáltatásoknak. (Lásd a forrásokat.) Az alapértelmezett lekérdezésformátum a `Boolean`. Ennél a keresés az indexben az összes alfanumerikus szóra történik, és négy belső jelölésű (infix) logikai műveleti jelet lehet használni: & (és), | (vagy), ! (tagadás) és - (mínusz). A rangsorolt mód minden kifejezést egyesít, majd visszaadja a lekérdezés alapján legérdekesebbeknek minősülő dokumentumokat. A *Xapian* formátumnál a *libferris* közvetlenül adja át további feldolgozásra a háttérszolgáltatásnak a lekérdezést. Jelenleg kizárólag a *Xapian* háttérrendszer képes az ilyen lekérdezések kezelésére.

EA indexelés

Az *EA* indexek hozzáadásának és lekérdezésének folyamata rendkívül hasonló a teljes szövegesekéhez. Az *EA* indexek a `feaindexadd` és a `feaindexquery` paranccsal kezelhetők, mindkettő fogadja a már megismert `-P /index_elérési_útja` kapcsolót.
Az *EA* indexek hangolására három átadott érték használható, ezeket az indexek létrehozásakor kell megadni. Azt szabják meg, hogy a fájlok indexelésekor mely *EA*-k jussanak szerephez. Létrehozhatunk például egy kisebb *EA* indexet, amely csak fájlneveket, -méreteket és néhány képjellemzőt tartalmaz, segítségével jól tudunk képekre keresni. Dönthetünk úgy is, hogy bizonyos *EA*-kat figyelmen kívül hagyunk, például azért, mert kiszámításuk időigényes lenne, vagy egyszerűen érdektelenek számunkra. Ha az indexet nem akarjuk a fájlok sértetlenségének ellenőrzésére használni, akkor az *MD5-* és az *SHA-1-*

kivonatok és az egyéb ellenőrző összegek elhagyásával számottevő mennyiségű időt takaríthatunk meg, ugyanis az ellenőrző összegek számításához végig be kell olvasni az összes fájlt.

Az *EA* indexekre vonatkozó általános átadott értékek közül az első a `max-values-size-to-index`, amely bájtokban mérve azt adja meg, hogy legfeljebb mekkora lehet a valamely jellemző vizsgálatakor az indexhez hozzáadott érték. A legtöbb *EA* értéke röviden, kevesebb mint 100 bájton ábrázolható. Az alapbeállítás meglehetősen bőkezű, és bármely *EA* értéknél 1024 bájtos maximális méretet engedélyez. A másik két átadott érték az `attributes-not-to-index` és az `attributes-not-to-index-regex`. Ezek a fájlok indexhez való hozzáadásakor figyelmen kívül hagyandó *EA*-k nevét adják meg. Természetesen magunknak kell eldöntenünk, hogy minden *EA*-t indexelünk, ekkor a lekérdezésekhez minden adat rendelkezésünkre fog állni, viszont a fájlok hozzáadása lassabb lesz, vagy csak az *EA*-k egy kisebb részét vesszük figyelembe, amivel felgyorsul a hozzáadás, ám a lekérdezések egy része eredménytelen lesz.

A `not-to-index` átadott értékek alapbeállításainak lehetővé kell tenniük a fájlok gyors hozzáadását, ugyanakkor biztosítaniuk kell a figyelemre érdemes *EA*-k indexelését. A három átadott érték alapbeállítását a `cc/capplets/index/ferris-capplet-index` segédprogram futtatásával lehet felülbírálni. A program megadja az új indexek létrehozásakor figyelembe vett alapbeállításokat, és a további fájlindexelésekhez módosítja a *~/.ferris* alatt található indexeket.

Az *EA* index tárolására *PostgreSQL* adatbázist használunk. Az *EA* indexek zökkenőmentes létrehozásához az `fcreate` eszközök futtatása előtt némi telepítési-beállítási jellegű munkát is kell végeznünk. A `PGSQL` nyelvet alapértelmezés szerint engedélyezni kell az új adatbázisokhoz. Rootként futtatva az alábbi parancs pontosan ilyen hatással jár:

```
# createlang -d sablon1 plpgsql
```

Ha nem akarjuk megváltoztatni a `sablon1` adatbázist, akkor a *PostgreSQL* adatbázist kézzel is létrehozhatjuk. Engedélyezzük hozzá a `plpgsql`-t, majd az alábbi `fcreate` parancshoz fűzzük hozzá a `db-exists=1` karakterláncot.

A *PostgreSQL EA* indexek esetében a *PostgreSQL* adatbázis által használt felhasználónevet, jelszót, állomásnevet, kaput és adatbázisnevet is megadhatjuk. Alapesetben (`db-exists=0`) a megadott nevű adatbázisnak még nem szabad léteznie, hiszen éppen most hozzuk létre az új *EA* indexhez.

A relációs adatbázisra alapuló *EA* indexekhez tartozik még egy érdekes beállítás, ennek segítségével megváltoztathatjuk, hogy bizonyos *EA*-k hogyan kerülnek normalizálásra az adatbázisban. Az alapértékek nagy valószínűséggel ebben az esetben is megfelelők. Röviden összefoglalom a kérdés lényegét.

Az `extra-columns-to-inline-in-docmap` megadja azon *EA*-k listáját, amelyek olyan fontosak a keresések szempontjából, hogy denormalizálni kellene őket a `docmap` táblába. A szinte minden fájl esetében egyedi értéket mutató EA-k tárolását hatékonyabban lehet megoldani a `docmap` táblában, belsőleg. Ha egy *EA*-t ilyen módon szeretnénk denormalizálni, akkor az *EA SQL*-típusát is meg kell adnunk.

1. táblázat *Példák EA-kra*

name-extension	A fájl kiterjesztése, például tar
treeicon	Egy a fájlhoz illeszkedő kép URL-je
is-audio-object	A fájl MIME főtípusa audio
is-source-object	A fájl forráskódot tárol
is-remote	A fájl a gép számára távoli.
language-human	A fájl emberi nyelve
a52-channels	Hangcsatornák száma
év	A zeneszámot tartalmazó album, lemez kiadásának éve

EA normalizálása relációs adatbázisban

A normalizált *EA*-kra vonatkozó lekérdezések teljesítésében négy tábla jut szerephez. A `docmap` tábla a fájl *URL*-jét és mesterségesen előállított belső kulcsát, a `docid`-t tárolja. Az `attrmap` tábla egy *EA* nevét tartalmazza, és szintén mesterséges belső kulcsot rendel hozzá, ez az `attrid`. Emellett az érték típusától függően a számos – egyébként rendkívül hasonló – `valuemap` tábla valamelyikét használjuk.

Az `strlookup` például a `varchar` értékekhez egy mesterséges belső kulcsot rendel, ez a `vid`. Végül, egy egyesítő (join) tábla, a `docattrs` egyesít egy `docid`-t egy `attrid`-vel és egy `vid`-vel, ezzel rögzítve, hogy adott fájl adott értékű jellemzővel rendelkezik. Ha tehát a (szélesség<=800) lekérdezést akarjuk feldolgozni, és a szélesség *EA* normalizálva van, akkor kell végeznünk egy keresést az `attrid`-re és a `vid`-re, egyesítenünk kell őket a `docattrs` táblába, ezzel előáll a lekérdezés feltételeinek megfelelő szélességgel rendelkező `docid`-k listája.

A normalizált *EA*-k tárolása egy oszlop formájában közvetlenül a `docmap` táblában történik. A fenti szélességre keresés végrehajtása során az egyező `docid`-k közvetlen keresése a `docmap.szélesség` oszlopról készített relációs adatbázis index alapján történik. A normalizálással időt takaríthatunk meg, miközben tárhelyet használunk fel. Általában elmondható, hogy a keresésekben gyakran szereplő *EA*-kat érdemes belsőleg tárolni. A *stat(2)* hívásokban nem szereplő vagy érdekesnek ítélt *EA*-k indexelését az `attrmap`, a `valuemap` és a `docattrs` táblára érdemes hagyni.

Én például saját felhasználónevemet használom, az adatbázis neve (`dbname`) pedig `lj`. Az alábbiakban a második paranccsal, a nem interaktív fcreate segédeszközzel létrehozom az *EA* indexet. A harmadik paranccsal a megosztott képkönyvtáramból az összes *JPEG* fájlt hozzáadom az indexhez. A `feaindexadd` parancsot is használhatjuk a `-d` kapcsolóval, ekkor pontosan felsorolhatjuk a parancssorban a kívánt elérési útvonalakat. A `-d` kapcsoló nélkül futtatva a `feaindexadd` rekurzívan megvizsgálja a megadott elérési utat:

```
$ mkdir /tmp/ea-index
$ fcreate --create-type=eaindexpostgresql \
--target-path=/tmp/ea-index dbname=lj user=ben
```

```
# ha már van új adatbázisunk, fűzzük hozzá
# a db-exists=1 kapcsolót
$ find /usr/share/backgrounds/images \
-name "*.jpg" \
| feaindexadd -P /tmp/ea-index --filelist-stdin
```

A képkönyvtáramban 42 JPEG fájl található. Kérdezzük le az indexet:

```
$ feaindexquery -P /tmp/ea-index '(width>=640)'
Found 34 matches at the following locations:
file:///usr/share/backgrounds/images/
➥dewdop_leaf.jpg
...
$ feaindexquery -P /tmp/ea-index '(size>=100k)'
Found 42 matches at the following locations:
file:///usr/share/backgrounds/images/
➥dewdop_leaf.jpg
...
$ feaindexquery -P /tmp/ea-index \
'(&(width<=800)(size>=100k))'
Found 19 matches at the following locations:
file:///usr/.../images/space/
➥apollo08_earthrise.jpg
...
```

Az *EA* index lekérdezésének írásmódja az *LDAP* keresési szűrők karakterlánc formátumú megadására épül (*„The String Representation of LDAP Search Filters", RFC 2254*). Meglehetősen egyszerű írásmódról van szó, a bal és a jobb oldalon szereplő érték összevetésére alkalmas kifejezéseket állíthatunk össze, ezeket kombinálhatjuk, illetve logikai ÉS (&), VAGY (|) és NEM (!) műveletekkel egészíthetjük ki. Minden kifejezés zárójelek közé kerül, a műveleti jelek argumentumaik előtt szerepelnek. A műveleti jelek fölöttébb egyszerűek: == az egyenlőség, <= és >= az értéktartomány és =~ a reguláris kifejezés egyeztetése.

Keresés a múltból

Az *ODBC* (kiegészítő szolgáltatásként) és a *PostgreSQL* (minden esetben) *EA* indexelő beépülő modulok lehetővé teszik, hogy egy fájl *EA*-inak több változatát is tároljuk. Ha egy fájl metaadataiból több változattal is rendelkezünk, akkor annak valamikor korábban érvényes jellemzői alapján is végezhetünk kereséseket.
A szolgáltatás használatához egy különleges *EA* segítségével meg kell adnunk a bennünket érdeklő időszakot. Az időkorlátozó *EA*-k az `atime`, a `ferris-current-time`, a `multiversion-mtime` és a `multiversion-atime`. A két utóbbi *EA* összehasonlítása a keresett fájl `mtime` és `atime` értékével történik. Az adott fájl indexadatainak egy változatához tartozó `ferris-current-time` *EA* a fájl indexelésének időpontját adja meg. Ha időtartományt nem választunk ki, akkor a lekérdezés csak az egyes fájlok metaadatainak legújabb változatát veszi figyelembe.
Az időkorlátozásokat karakterlánc formájában adhatjuk meg, a *libferris* megpróbálja a lehető legpontosabban meghatározni a kapott karakterlánc formátumát. A *libferris* csomag *tests/timeparsing* könyvtárában található egy időfeldolgozó segédeszköz, ennek időértékeket tudunk átadni, és így ki tudjuk deríteni, hogy a *libferris* mit hoz ki az egyes karakterláncokból. A használható időkarakterláncokról a *libferris GYK* vonatkozó részében találunk bővebb ismertetőt. (Lásd a forrásokat.)
Az alábbi példa az időkorlátozott lekérdezést szemlélteti; ebben az esetben az egy évvel ezelőtt indexelt, meghatározott szélességtartományba eső képeket keressük:

```
$ feaindexquery -P /tmp/ea-index \
 '(&(width>=1600)(ferris-current-time<=1
 ➥year ago))'
```

Ha egy nagyobb fájlt két éve indexeltünk, majd később lecseréltük a kicsinyített változatára és újraindexeltük, akkor szerepelni fog a fenti lekérdezés eredményében. Ennek oka az, hogy metaadatainak egyik változata egyezést mutat a keresési feltételekkel.
Mivel az *EA* lekérdezésekre vonatkozó időkorlátozások kezelése ugyanazon a felületen keresztül történik, mint az *EA* értékekre vonatkozóké, a kívánt időtartományt a megszokott lekérdezési módszerekkel tudjuk megadni. Kiválaszthatjuk például azokat a dokumentumokat, amelyek indexelése 2003-ban és megadott szélességgel történt, vagy a megadott személy tulajdonában lévő, az elmúlt hónapban módosítottakat:

```
## megjegyzés: egy sorban az egész
$ feaindexquery -P /tmp/ea-index '
(|
  (&
    (width>=1600)(ferris-current-time>=begin
    ➥2003)
    (ferris-current-time<=end 2003)
  )
  (&
    (owner-name==sarusama)
    (multiversion-mtime>=end last month)
  )
)
```

Összefoglalás

Őszintén remélem, hogy sikerült érthetően felvázolnom, mire is képes a *libferris* jelenlegi megvalósítása az *EA* alapú és a teljes szöveges indexelés terén, és minél több olvasó figyelmét sikerült felkeltenem. A jelenlegi változat elkészítése szükségszerű lépés volt egy sokkal nagyobb mértékben formalizált szemantikus fájlrendszerlekérdező és böngésző felület megvalósításának végső célja felé.

Linux Journal 2005. február, 130. szám

A cikkhez tartozó források elérhetősége:
➲ *www.linuxjournal.com/article/7928*

Ben Martin több mint tíz évig fájlkezelőkön dolgozott. Jelenleg doktoranduszként a jelentéstani fájlrendszerek és a formális fogalomanalízis egyesítésén dolgozik, amitől az ember-fájlrendszer kapcsolattartás javulását várja.

Merevlemez nélküli linuxos X terminál

Hogyan indítsunk hálózatról és használjuk meghajtó nélküli X terminálként állandó háttértár nélküli Linuxot?

Az *X* terminál nem igazán új ötlet; az *NCD* és más cégek legalább 15 éve gyártanak ilyeneket. A vékony ügyfél ötlete az 1990-es években kezdett igazán divatba jönni, amikor a PC alkatrészek ára olyan mélyre zuhant, hogy nem volt tovább pénzügyi jelentősége az *X* terminálóknak. Komoly viták kezdődtek a teljes bekerülési költségről (amibe az alkatrész és a karbantartási költségeket is beleszámolták) a vékony ügyfelek és a PC támogatói között, de ezt a vitát ez a cikk sem fogja eldönteni. Célunk mindössze annyi, hogy megmutassuk, hogyan lehet a rohamléptekkel fejlődő PC technológia nyomán hátramaradt elavult alkatrész halmainkból *X* terminálokat építeni.

Minden vékony ügyfél legfontosabb tulajdonsága, hogy egyáltalán nincs, vagy csak nagyon kevés állandó tára van. Általában az elve *X* terminálnak szánt terminálokban kis mennyiségű *NVRAM*-ot alkalmaznak, ahol a beállítási adatokon kívül semmi mást nem tárolnak. Gyakorlatban ezeket a az adatokat is áttehetjük a kiszolgálón tárolt beállításállományba, amit a terminál induláskor letölthet. Ebben a cikkben azt a letisztult elvet követjük, miszerint egy *X* terminálnak egyáltalán nincs szüksége tárhelyre.

PXE Indítás

A PC-nek nincs merev, hajlékony vagy CD lemeze úgyhogy az indítóprogram és az indítható állomány tárolásához valamilyen más eszközt kell találnunk. Az *X* terminálok a lakhelyükül szolgáló hálózat teremtményei, így magától értetődő választás lenne a hálózati csatolófelület kártya (*NIC*). Ennek megvalósításához a *NIC*-nek rendszerindító eszközként kell regisztrálnia magát a *BIOS*-ban. Ha kiválasztják, le kell tudnia tölteni a rendszerindítót a hálózatról. Ez nem igazán olyasmi, amit minden futószalagról lekerült *NIC* csuklóból tudna. Az *Intel* ugyanakkor kiadta a *NIC*-hez szánt indító *ROM*-ját, amelyet *PXE*-nek (*Preboot eXecution Environment,* kiejtve*: pixie*) keresztelt el, s amit aztán a cég és számos más gyártó be is épített bizonyos termékekbe. Sok újabb, beépített hálózati kártyával rendelkező alaplapban találunk *PXE* támogatást.

A cikkre készülés során öt különféle NIC-et teszteltem amelyek a hirdetések szerint támogatják a *PXE*-t: az *Intel PRO/100+ (PILA8460BNG1)*, a *3Com 3C905CX-TX-M*, a *D-Link DFE-550TX*, a *Linksys LNE100TX* és az *SMC 1255TX* (*Tulip* lapkakészlet) lapkát. Az ötből kizárólag a *3Com* kártya volt képes minden további nélkül elindulni. Az *SMC* kártyához tudtam külön szerezni indító *ROM*-ot így végül az is működött. A másik három kártyán feltűnő, ám üres foglalatot találtam az indító *ROM* helyén, amit alapértelmezés szerint nem szállítanak. Nem árt az óvatosság vásárláskor.

Amikor az alaplap *BIOS*-a a *PXE NIC BIOS*-t választja indítóeszközként, az kiad egy *DHCP* kérelmet a *LAN*-on, majd a visszakapott válaszokban a *PXE* kiterjesztéseket keres. Ha ilyen kiterjesztést tartalmazó választ talál, akkor visszajelez és elfogadja a választ. A kiszolgáló válaszában különösen a `next-server` és a `filename` paramétereket keresi. Ezek a paraméterek adják meg a *TFTP* kiszolgáló IP számát valamint az ügyfél által letöltendő és elindítandó rendszerbetöltő állomány nevét.

DHCP és TFTP

Az *Internet Software Consortium* 3.0-ás verziójú *DHCP* kiszolgálóját *PXE* kiterjesztések kezelésére is be lehet állítani. Szerencsére ezt a *DHCP* kiszolgálót kapjuk a legtöbb *Linux* terjesztéssel, ide értve a *Red Hat 8.0* és későbbi verzióit. Az *1. listában* egy olyan *DHCP* kiszolgáló beállításállományát (*dhcpd.conf*) olvashatjuk, amely *PXE* kiterjesztésekkel bővített *DHCP* válaszokat küld, amennyiben a *DHCP* ügyfél *PXE NIC* néven azonosítja magát.

Az itt olvasható beállítások szerint az ügyfél a *192.168.1.1* címen található *TFTP* kiszolgálóról tölti le a pxelinux.0 állományt. A beállításfájlban felsorolt kapcsolók listáját az *1. táblázatban* találjuk.

A `192.168.1.1` címen található kiszolgálónkat természetesen úgy kell beállítani, hogy képes legyen *TFTP* szolgáltatást nyújtani. Ezen kívül kell lennie rajta egy `pxelinux.0` rendszerbetöltő állománynak, mégpedig ott, ahol a *TFTP* kiszolgáló folyamatok keresik (általában a */tftpboot* könyvtárban). A *TFTP* kiszolgáló folyamatot általában valamelyik szuperkiszolgáló kezeli (*inetd* vagy *xinetd*) következésképpen ha be szeretnénk kapcsolni valamelyikük beállításfájljában (*/etc/inetd.conf* vagy */etc/xinetd.conf*) kell matatnunk.

A `pxelinux.0` nevű rendszerbetöltő állomány *H. Peter Anvin SYSLINUX* projektjéből származik. Az általános

1. táblázat *A dhcpd.conf állományban használható PXE-vonatkozású kódok leírása*

Code	Meaning
1	Az indító fájlkiszolgáló multicast IP címe.
2	UDP kapu melyen az ügyfél az MTFTP válaszokat figyelheti.
3	UDP kapu melyet az MTFTP kiszolgálók MTFTP kérelmek fogadására használnak.
4	Az ügyfélnek ennyi másodpercet kell aktivitásra várakoznia mielőtt új MTFTP átvitelt kezdeményezne.
5	Az ügyfél ennyi másodpercet vár, mielőtt újraindítaná az MTFTP átvitelt.

rendszerbetöltőkkel szemben mint amilyen a ***LILO*** vagy a ***GRUB***, a ***PXELINUX*** képes a ***PXE*** protokollt kezelésére és el tudja látni a szükséges hálózati feladatokat, így ezen a ponton már képes átvenni a rendszerindítást és ***TFTP***-n keresztül letölteni a rendszermagot valamint a tömörített ***RAM*** lemezt. Ugyanakkor a ***PXELINUX*** használatához bővített ***TFTP*** kiszolgálóra lesz szükségünk, amelyik képes megérteni a `TSIZE` kapcsolót (***RFC 2349***). Szerencsére, ***H. Peter Anvin*** a szabványos ***BSD TFTP*** démon ***tftp-hpa*** névre keresztelt, javított verzióját is elkészítette amelyik már képes e parancs értelmezésére. Könnyű dolgunk van: a ***/usr/sbin/in.tftpd*** könyvtárban található ***TFTP*** démont egyszerűen csak lecseréljük a ***tftp-hpa*** programra.

PXELINUX

A ***PXELINUX*** tudja, hogy a ***PXE*** indító ***ROM*** hová teszi a ***DHCP*** kiszolgálótól kapott hálózati paramétereket a memóriában, így képes felhasználni őket egy új ***TFTP*** folyamat megindításához amellyel saját beállításállományait tölti le a kiszolgálóról. A fentiek szerint beállított ***TFTP*** kiszolgálón az ügyfélen futó rendszerbetöltő először a ***/tftpboot/pxelinux.cfg/ethermac*** könyvtárban próbálja megtalálni a beállításállományait. Az ***ethermac*** szó itt az ügyfél kisbetűs hexadecimális kódban megadott Ethernet hardvercímének felel meg, ahol a nyolcadokat kötőjelek választják el, például: ***fe-ed-de-ad-be-ef***. Ha ezt nem tudja elérni, a rendszertöltő ***/tftpboot/pxelinux.cfg/iphex állományt keresi, ahol az iphex*** az ügyfél IP címe, nagybetűs hexadecimális kóddal megadva. Például, ha az ügyfél ***IP*** címe ***192.168.0.12***, a ***PXELINUX*** a ***/tftpboot/pxelinux.cfg/C0A8000C*** állományt keresné. Amennyiben ez a fájl sem létezik, a legkevésbé lényeges szakaszt kivágja a névből és újrakezdi a folyamatot. Így a fenti példában ha a ***C0A8000C*** állományt nem találja, a ***PXELINUX*** megpróbálkozik a ***C0A8000***, majd a ***C0A800*** állományokkal és így tovább. Ezáltal egyetlen beállításállományt használhatunk az egész alhálózathoz, feltéve hogy az alhálózat határait a szakaszokhoz igazítottak. A 2. *lista* a ***PXELINUX*** beállításállomány tartalmát mutatja be. Az első sor a letöltendő tömörített rendszermag állomány nevét adja meg. Minden elérési út a kiszolgáló

1. lista PXE ügyfeleket is támogató dhcpd.conf állomány minta

```
option space PXE;
option PXE.mtftp-ip
  code 1 = ip-address;
option PXE.mtftp-cport
  code 2 = unsigned integer 16;
option PXE.mtftp-sport
  code 3 = unsigned integer 16;
option PXE.mtftp-tmout
  code 4 = unsigned integer 8;
option PXE.mtftp-delay
  code 5 = unsigned integer 8;
option PXE.discovery-control
  code 6 = unsigned integer 8;
option PXE.discovery-mcast-addr
  code 7 = ip-address;
subnet 192.168.1.0 netmask 255.255.255.0 {
  class "pxeclients" {
    match if substring (option
      vendor-class-identifier, 0, 9) =
      ➥"PXEClient";
    option vendor-class-identifier "PXEClient";
    vendor-option-space PXE;
    # Legalább az egyik gyártófüggő PXE
    # kapcsolót be kell kapcsolnunk
    # hogy az ügyfél
    # indító ROM-ja felismerje, hogy PXE-kezelő
    # kiszolgálóval van dolga.
    # Mi a MCAST IP címet
    # állítottuk 0.0.0.0-ra jelezve
    # a ROM-nak, hogy
    # nem kezelünk multicast TFTP-t.
    option PXE.mtftp-ip 0.0.0.0;
    # Itt adjuk meg annak a fájlnak nevét
    # amit a ROM-ok
    # letölthetnek.
    filename "pxelinux.0";
    # A kiszolgáló neve amiről le kell
    # tölteniük.
    next-server 192.168.1.1;
  }
  pool {
    max-lease-time 86400;
    default-lease-time 86400;
    range 192.168.1.2 192.168.1.254;

   # Amennyiben ezt a sort beletesszük,
   # minden ügyfélhez
   # külön gépbejegyzéseket kell
   # készítenünk, ahol az
   # ethernet MAC címeket a IP címekkel
   # összerendelhetjük.
   # deny unknown clients;
  }
}
```

2. lista A PXELINUX beállításállománya határozza meg melyik tömörített rendszermag fájlt kell letölteni

```
DEFAULT vmlinuz
APPEND initrd=ramdisk.gz ramdisk=65536
root=/dev/ram rw
IPAPPEND 1
```

/tftpboot könyvtárához képest relatív kell legyen. A második sorban a rendszermagnak átadandó paramétereket soroljuk fel, jelezve, hogy a gyökér fájlrendszert 64MB-os memória lemezre kell felcsatolni írható/olvasható módon. Az utolsó sor hatására a ***PXELINUX*** egy további rendszermag paramétert hoz létre:

```
ip=client-ip:server-ip:gateway-ip:netmask
```

Ehhez azokat az adatokat használja fel, amit a ***PXE*** indító ***ROM*** a ***DHCP*** kiszolgáló válaszában kapott meg. Ez akkor lehet hasznos, ha a rendszermagot engedélyezett rendszermag-szintű IP hálózattal fordítottuk. Ilyenkor ugyanis a rendszermag ezeket a paramétereket felhasználja a hálózati csatolófelület beállításához, így később ezt már nem kell megtennünk az indítóparancsfájlban az `ifconfig` vagy `ifup` futtatásával.

Rendszermag készítése

A rendszermag szintű automata ***IP*** paraméter beállítás használatához a hálózati meghajtóeszköznek viszonylag hamar elérhetőnek kell lennie, még a root fájlrendszer felcsatolása előtt. Következésképpen nem tölthetjük be modulként. Minthogy a legtöbb terjesztéssel együtt érkező rendszermagok igen kiterjedten használnak modulokat, gyakorlatilag kénytelenek vagyunk új rendszermagot forgatni az ***X*** terminálunkhoz. Továbbá a saját rendszermagunknak kezelnie kell a ***RAM*** lemezeket és a indítási ***RAM*** lemezeket. A rendszermag szintű ***IP*** hálózat automata beállítása szintén kényelmes dolog. Persze nem kötelező semmilyen dinamikus módszert alkalmaznunk az ***IP*** címek megadásához (választhatjuk a ***DHCP***, ***BOOTP*** vagy ***RARP*** megoldásokat is), a ***PXELINUX*** beállításfájljában megadott ***IPAPPEND*** viszont biztosítja számunkra, hogy rendszermag biztosan a megfelelő IP paramétereket kapja meg. Végül ha az eszköz fájlrendszert automatikusan csatoljuk a ***devfs*** segítségével, jelentősen leegyszerűsíthetjük a RAM-gyökérlemezen található */dev* könyvtárunkat.

A RAM lemez root fájlrendszere

Egy lemez nélküli ***Linux X*** munkaállomás esetében legnehezebb feladatunk a gyökér fájlrendszer kialakítása és feltöltése lenne, ha nem jelent volna meg ***Richard Gooch*** eszköz fájlrendszere és Erik ***Anderson BusyBox*** kombinált végrehajtható állománya. Az eszközkezelő fájlrendszer automatikusan kezeli a */dev* könyvtárat, a modulok rendszermagba töltésekor szükség szerint hozza létre a megfelelő eszközvégpontokat. Ez két dolgot jelent: a könyvtárban nincsenek szükségtelen bejegyzések, és a ***RAM*** lemezes fájlrendszer készítőinek nem kell órákat töltenie az `mknod`-al, hogy létrehozza az összes szükséges végpontot. A ***BusyBox*** egyesített program olyan végrehajtható állomány amely indítástól függően változtatja személyiségét. Szokásos használata során a */bin/busybox* programra mutató közvetett hivatkozásokat hozunk létre a */bin/ls*, */bin/cat*, */bin/ps*, */sbin/mount* és a egyéb neveken amivel egy minimalista ***UNIX*** rendszerhez jutunk. Semmilyen egyéb futtatható állományra vagy programkönyvtárra nincs szükségünk; a ***BusyBox*** már mindent tartalmaz.
Úgy is fogalmazhatnánk, hogy az eszközfájlrendszer felügyeli a */dev* könyvtárat, a ***BusyBox*** a */bin* és */sbin* könyvtárakat; a rendszermag kezeli */proc* mappát; a csak olvasható ***NFS*** csatolás pedig az */usr* könyvtárért felelős; a */tmp* üres marad. Így egyetlen dologgal kell csak törődnünk, az */etc*-vel. Szerencsére az */etc* könyvtárba se kell túl sok minden, elég ha tartalmazza az */etc/fstab*, */etc/inittab* és */etc/init.d/rcS* állományokat, ez utóbbi az az indító-parancsfájl amit a ***BusyBox*** `init`-ként futva használ.
A ***BusyBox*** a beágyazott ***Linux*** világnak íródott és általában statikus futtatható állományként fordítják. ugyanakkor, az ***XFree86*** kiszolgáló önmagában is sok, a */lib* könyvtárban található osztott könyvtártól függ. A */usr*-t hálózati fájlrendszerként (***NFS***) csatoljuk, tehát az */usr/lib*-ben található osztott könyvtárak miatt nem kell aggódnunk, biztosítanunk kell viszont azokat, amelyeket a */lib*-ben keres majd az ***XFree86***. Emiatt érdemes kihasználhatunk egy helymegtakarítási lehetőséget, nevezetesen, hogy a ***BusyBox***-ot dinamikus végrehajtható állományként állítjuk be és fordítjuk le. Az ***XFree86*** futtatásához szükséges minimális */lib* programkönyvtárak kigyűjtését az

```
ldd /usr/X11R6/bin/XFree86
```

parancs kiadásával tudjuk elvégezni. Ezek a ***glibc*** (***libc.so*** és ***libm.so***), a PAM (***libpam.so*** és ***libpam_misc.so***) valamint maga a dinamikus töltő (***libdl.so*** és ***ld-linux.so***) lesznek.

XFree86 beállítása

Az ***XFree86*** végrehajtható állomány általában az */usr/X11R6/bin* könyvtárban található, amely a */usr* alkönyvtára. Így az ***X*** kiszolgálót nem kell ***RAM*** lemezünkön tartanunk, ehelyett az ***NFS*** csatolásról betölthetjük. Bár a moduláris ***XFree86*** kiszolgáló a 4.0-ás verzió óta már nem alkatrészfüggő, a beállításállománya bizony igencsak az. Amennyiben több különféle videó alkatrésszel rendelkező gépet kezelünk, nem használhatjuk valamennyi géphez ugyanazt az ***XF86Config*** állományt. Ezért inkább nem tartjuk a gyökér fájlrendszer ***RAM*** lemezén, ahol általában egyébként */etc/X11/XF86Config* néven szerepel. Ehelyett inkább terminálonkénti beállításállományokat tárolunk az */usr NFS* könyvtárban. Végül, a ***BusyBox*** `init` folyamatát úgy állítjuk be, hogy egyetlen sort tartalmazó parancsfájlt indítson újra folyamatosan:

```
/usr/X11R6/bin/XFree86 \
-xf86config /usr/X11R6/configs/iphex -query \
server
```

3. lista A terminálok XF86Config állományának részlete

```
Section "Files"
        RgbPath         "/usr/X11R6/lib/X11/rgb"
        ModulePath      "/usr/X11R6/lib/modules"
        FontPath        "tcp/192.168.1.1:7100"
EndSection
```

ahol az `iphex` az ügyfél ***IP*** címe hexadecimális kódban (a ***PXELINUX***-tól kölcsönzött jelölési rendszer szerint) a `server` pedig kiszolgáló ***IP*** címe ponttal elválasztott decimális kódban. Néhány ügyes ***awk*** trükk a ***/proc/cmdline***-on és teljesen elkerülhetjük a gépnevek és ***IP*** címek ***RAM*** lemezbe kódolását.
Alapszintű ***XFree86*** beállításállományt az `XFree86 -configure` parancs kiadásával hozhatunk létre a terminálon. Ez többnyire helyesen ismeri fel a videó alkatrészt az eredményül kapott beállításállomány pedig betölti a megfelelő ***XFree86*** modulokat. Érdemes megemlíteni azonban, hogy az alapértelmezett mutató eszköz a /dev/mouse lesz, ami általában nem szerepel az eszköz fájlrendszerrel létrehozott eszközök között. Például, a ***PS/2***-es egér nem itt, hanem a ***/dev/misc/psaux*** helyen található meg.

Beállítások a kiszolgálón

Amitől az ***X*** terminálunk tényleg ***X*** terminál lesz egyszerű grafikus kijelzővel felszerelt ***Linux*** gép helyett az a fenti ***XFree86*** parancssorunk `-query` ***server*** része. Hatására a terminálon futó ***XFree86*** kiszolgáló elindítja az ***XDMCP (X Display Manager Control Protocol)*** folyamatát és egy olyan kiszolgálót keres, amely kezelhetné a képernyőjét. Csakhogy erre nem minden kiszolgáló kapható.
Először is, a kiszolgálónak értelemszerűen hallgatnia kell a bejövő ***XDMCP*** kapcsolatokra. Az ***XDMCP*** általában a 177-es ***UDP*** kaput használja, és a legtöbb megjelenítésvezérlő (***xdm***, ***gdm***, ***kdm***) be is állítható úgy, hogy engedélyezze az ***XDMCP*** kérelmeket. Bár a legtöbb terjesztés alapértelmezés szerint grafikus felülettel indul, biztonsági megfontolásokból szinte mindegyik tiltja a bejövő ***XDMCP*** kérelmeket. Például a klasszikus ***xdm X*** megjelenítőkezelő beállításállományában (amely általában a ***/etc/X11/xdm/xdm-config***) a következő sort találjuk alapértelmezés szerint:

```
DisplayManager.requestPort: 0
```

Ezt a sort megjegyzésbe kell tennünk, ha azt szeretnénk, hogy az ***xdm*** fogadni tudja az ***XDMCP*** kérelmeket. Ezen felül az ***xdm*** beállítható úgy is, hogy gépenkénti vagy alhálózat alapon korlátozza saját elérhetőségét, amit a ***/etc/X11/xdm/Xaccess*** beállításállományban tehetünk meg(senkit ne zavarjon meg a ***/etc/X11/xdm/Xservers***, ami lényegében történelmi emlék). Például, ha az ***xdm***-et a ***192.168.1.0/24*** alhálózat termináljaira szeretnénk korlátozni, mindössze a `192.168.1.0/24` sort kell felvennünk a ***/etc/X11/xdm/Xaccess*** állomány végére.
Ezen felül jól jöhet, ha a kiszolgáló az ***xfs X*** fontkiszolgálón keresztül fontokat is tud nyújtani a terminálóknak.
Akárcsak az előbbi esetben, a legtöbb terjesztés saját fontkiszolgáló folyamatot futtat, mégis általában úgy van beállítva, hogy nem fogad bejövő kérelmeket. Például a az ***xfs*** beállításállományában, az ***/etc/X11/fs/config***-ban, általában a `no-listen = tcp` sort találjuk. Amennyiben ezt megjegyzésbe helyezzük, a terminálok ***XF86Config*** állományának (amelyet a kiszolgáló ***/usr/X11R6/configs/iphex*** állományában találunk) ***Files*** szakasza mindössze egyetlen ***FontPath*** bejegyzést kell tartalmazzon a szokásos fél tucat helyett, amint azt a *3. listában* olvashatjuk (ahol egyébként feltételezzük, hogy a kiszolgáló IP száma ***192.168.1.1***).
Végül a kiszolgálót be kell állítanunk, hogy támogassa ***/usr*** fájlrendszerének csak olvasható az ***NFS*** megosztását a terminálok részére, hiszen ez az a hely ahol a terminálok ***XFree86*** kiszolgálójukat keresik.

Néhány szó a biztonságról

Az ***X*** terminálok futtatása során számos biztonsági kérdésre kell ügyelnünk. Először is, az ***xdm*** és ***xfs*** beállításokban végzett változtatásaink elég egyértelműen visszafordítottak bizonyos dolgokat amit egyébként a kiszolgáló biztonsága növelése érdekében tettek. Ezen felül, a cikkben leírt változat semmilyen forgalmat sem titkosít. A terminál minden billentyűleütése kódolatlanul megy keresztül a hálózaton. Az ***X*** teminálokkal dolgozó egyetlen biztonságosnak mondható módszer, ha valamennyit olyan privát ***LAN*** hálózatra helyezzük, amelyet kizárólag az ***X*** terminálok használnak és nem lehet belőle az Internet-re kijutni. A terminálok és a kiszolgáló egyetlen kártyáján kívül senki más nem csatlakozhat erre a ***LAN***-ra.

Letölthető készletek

A nyomtatott anyag korlátozott mérete miatt ez a cikk csak magas szintű áttekintést nyújthatott a Linux gépek lemez nélküli indításáról és ***X*** terminállá alakításáról, így nem mentünk bele a pontos megvalósítás apró részleteibe. Az érdeklődő olvasók a szerző honlapjáról letölthetik az ***X Terminal Kit*** készletet. A készletben parancsfájlokat, ***Makefile*** állományokat és ***README***-ket találhatnak amelyek végigvezetnek a néha kicsit bonyolulttá váló folyamaton. Ezen felül a cikkben bemutatott program az Internet különféle helyeiről származik. Valamennyi hely igen részletes leírással rendelkezik saját csomagjait illetően. További útmutatást a hálózati forrásokban találunk.

Linux Journal 2005. február, 130. szám

A cikk forrásai:
➲ ***www.linuxjournal.com/article/7924***

Chip Coldwell
(coldwell@physics.harvard.edu)
A Harvard Egyetem Fizika Tanszékének rendszeradminisztrátora. Ha éppen nem valamilyen számítógéppel játszadozik, általában biciklin találjuk, vagy menyasszonyának, Cindynek társaságát élvezi.

Bloglines Webszolgáltatások

Egyre több weblap ajánlja fel szolgáltatásainak gépbarát verzióját. Az ilyen egyszerű de hasznos szolgáltatásra jó példa az új weblap tartalmakról küldött frissítés.

Az elmúlt hónapban azokat a módszereket vizsgáltuk amelyekkel begyűjthetjük vagy összerendezhetjük a különböző weblapok tartalmát majd a nap híreiből egyetlen összefoglalót készíthetünk. Bár izgalmas volt látni, mire jutunk mindössze pár sornyi kóddal, az általam bemutatott alkalmazás egyszerű kis játék a tényleges összesítőkhöz képest. A mintaalkalmazás egyetlen felhasználót kezel, primitív beállításállományon alapszik, nem csoportosítja a blogokat, csak akkor frissíti a weblogokat, amikor kifejezetten utasítjuk rá. Ezen kívül nem figyeli és kezeli a hibákat sem.

Figyelemebe véve a szükséges technikai és tervezési kérdéseket egy igazán felhasználóbarát, komoly gyűjtőprogram megírása már meghaladná cikkünk kereteit. Azonban néhány nappal az előtt, hogy leültem volna megírni ezt a cikket, valami lenyűgöző dolog történt. Az ingyenes ***Bloglines.com*** gyűjtőszolgáltatás, amit igen sok ember használ saját kedvenc weblogjainak összegyűjtésére, elérhetővé tette webszolgáltatás-programozási felületét, melynek segítségével független fejlesztők használhatják a ***Bloglines*** adatait és alkalmazásait saját alkalmazásaik létrehozásához és telepítéséhez. A ***Bloglines API*** megjelenése jól mutatja a webszolgáltatások növekvő népszerűségét az nagyobb honlapok között és ajtót nyit az új ***Bloglines*** infrastruktúrára épülő új alkalmazások előtt.

E hónapban a ***Bloglines API***-t vesszük szemügyre majd létrehozunk egy rá épülő egyszerű alkalmazást. Az ***API*** a cikk írásakor (2004 október eleje) teljesen újdonságnak számított és nyilvánvalóan fejlődni fog ahogy egyre több ember használja. Azok számára akiket érdekelnek a weblogok és eddig csak a webszolgáltatások praktikus felhasználási lehetőségeire vártak, az események ilyen fordulata éppen időben történt.

Mi is az a webszolgáltatás?

A webszolgáltatások mögött rejlő alapötlet nagyon egyszerű: A web sikerének titka nem kis részben annak köszönhető, hogy az ügyfél és a kiszolgáló operációs rendszere nem lényeges. Tehát amíg az ügyfél és a kiszolgáló is tartja magát, a ***HTTP*** és ***HTML*** szabályokhoz zökkenőmentesen tudnak beszélgetni egymással. A ***Linux*** is éppen ez által volt képes helyet biztosítani magának a kiszolgáló piacon.

A webszolgáltatások még egy lépéssel még tovább mennek, és azt mondják, hogy a web legnagyobb használói nem az emberek hanem a számítógépek. Bár a számítógépek HTTP-n keresztül kommunikálnak, az adatokat ***XML jelölőnyelven (markup language)*** vagy metanyelven küldik és fogadják, amely robbanásszerűen terjed az utóbbi években. Ha az én számítógépem képes ***XML***-t küldeni a számítógépednek küldött ***HTTP*** kérelemben, a géped pedig ***XML***-t ad vissza a HTTP válaszában, operációs rendszertől és nyelvtől függetlenül tudunk információt cserélni.

A szolgáltatás ***XML-RPC*** néven ismert eredeti formája még ma is létezik és kiválóan használható a gyors könnyű kapcsolattartásra. Az ötletet azóta továbbfejlesztették és rengeteg fajta adattípus, hibakezelő módszer és objektum sorbafejtő megoldás jelent meg, amelyek az ***XML-RPC*** nyelvből hiányoztak. Ez a kiterjesztést ***SOAP***-nak ***(Simple Object Access Protocol)*** nevezték el. A ***SOAP*** elméletileg sokféle protokoll felett képes futni, de a legtöbbször HTTP felett küldik.

A ***SOAP*** sok problémára kiváló megoldás, eltekintve attól, hogy iszonyú bonyolult, elég lassú és nehéz megvalósítani. Az ***XML-RPC*** és a ***SOAP*** egyaránt megköveteli, hogy a ***HTTP*** üzenetben a lekérdezést hibátlan ***XML*** alakban adjuk meg. A növekvő összetettségre válaszul született meg a ***REST (representational state transfer)***, ahol minden műveletet egyszerű ***HTTP GET*** kérelmek indítanak és az összes paramétert maga az ***URL*** tartalmazza. Válaszként a kérelemnek megfelelő bejegyzéseket és mezőket tartalmazó ***XML*** dokumentumot kapjuk vissza. A ***Bloglines API*** valamennyi hívását ***REST***-en keresztül végezzük, bár nehéz megmondani, hogy a fejlesztők a viszonylag egyszerű lekérdezések vagy a tervezési irányelvek miatt döntöttek így.

Bár a webszolgáltatások nem sokáig maradnak a cégek zárt ajtói mögött, csak néhány nagyobb webhely tette nyilvánossá terveit és ***API*** felületét. A legismertebb példák a web legnagyobb és legjövedelmezőbb vállalkozásai az ***Amazon***, az ***eBay*** és a ***Google***. Az ***eBay*** éves díj és tranzakciónkénti díj formájában pénzt kér webszolgáltatásainak eléréséért. Ezzel ellentétben az ***Amazon*** és a ***Google*** ingyenesen elérhetővé tették felületüket, igaz felhasználhatósági megkötésekkel és a későbbi elérhetőség ígérete nélkül.

Az ***API*** nyilvánossá tételével a ***Bloglines*** is jelzi, hogy az ***Amazon***, ***Google*** és ***eBay*** által létrehozott fejlesztői közösséghez hasonló tábort szeretne felállítani. Lépésével egyúttal azt is jelzi, hogy továbbra is szeretne a világ első weblog-gyűjtő és alkalmazás rendszere maradni. Tekintve, hogy

a *Google* megvásárolta a *Blogger*-t néhány évvel ezelőtt, valamint ha megnézzük, milyen széleskörű keresési megoldásokat kínál *API* felületén a *Bloglines*, lehet hogy egy új alkalmazás csatának lehetünk szemtanúi a figyelemért versengő *Google* és a *Bloglines* felület között.

A Bloglines API

A *Bloglines* nagy számú weblogból és gyakran frissített hírforrásokból gyűjt adatot. A *Bloglines* több különféle formátumban is szívesen fogad adatokat, ideértve az *Atom* és az *RSS* néhány verzióját. Sőt, a *Bloglines* megkérdezi feliratkozótól, hogy melyik típust szeretnék használni amennyiben egynél több is elérhető. A *Bloglines* ezután archiválja a tartalmat, keresőfelületet biztosítva az érdeklődőknek. A *Bloglinesban* néhány fontossági képességet is találunk, amelyek felhívják a feliratkozók figyelmét rá, hogy mely további weblogok érdekelhetik őket. Végül a *Bloglines* lehetővé teszi, hogy más felhasználók feliratkozásába is belekukkantsunk; ha valaki kíváncsi rá, mely weblogok érdekelnek engem, elolvashatja a adatlapomat és megtekintheti a feliratkozásaimat.
Egyelőre legalábbis, ezen funkciók nagy része még nincs leleplezve így csak a *Bloglines* weblap felületén keresztül érhetők el. Három képesség azonban már a *Bloglines Webszolgáltatások API*-ból is elérhető:

- *Figyelemfelhívás (Notifier)*: ha *Bloglines* feliratkozóként tudni szeretnénk, mikor érkezett új hír egy vagy több kiválasztott weblogunkra, ezzel a képességgel megtudhatjuk. Ez a *Bloglines Webszolgáltatások* legmegalapozottabb része, számtalan eszköz több különféle operációs rendszer és ablakozó eszközkészlet alá áll rendelkezésünkre amelyek ezt a felületet használják a frissítésekhez.
- *Szinkronizálás (Sync) API*: ezzel egy adott felhasználó feliratkozásairól gyűjthetünk információkat, a feliratkozások legfrissebb változataival egyetemben. Ezt úgy képzelhetjük el, mint a *Bloglines* által létrehozott fő weblog lista *HTML* kódja mögötti található adatokat.
- *Blogroll API*: módszer, amellyel a lekérhetjük és megjeleníthetjük egy adott felhasználó feliratkozási listáját.

Figyelemfelhívó API

Mint korábban írtam, a *Bloglines* a *REST* rendszerét használja az összes webkiszolgáló *API*-jához. Ennek megfelelően minden lekérés egyetlen *URL*-ből áll, ahol az összes paraméter és azok értékei az *URL*-be kódolva találhatók.
A visszakapott információ olyan alakú, amilyet a kiszolgáló megfelelőnek talál. Ez szöges ellentétben áll a *SOAP* megközelítésével, ahol minden egyes paraméter és visszaadott érték nevét és típusát előre meg kell határoznunk. A szabály alól egy kisebb kivétel találunk; amikor ugyanis az *API* azonosítás céljából felhasználónevet és jelszót kér, az alap *HTTP*-n keresztül és nem az *URL*-ben érkezik. A *Bloglines* világában, a feliratkozókat e-mail címük és felhasználó által választható jelszavuk azonosítja.
A legkönnyebben megérthető és használható *API* a *Notifier*. A *Notifier* meghívásához egyszerűen csak nyissuk meg

a *rpc.bloglines.com/update?user=reuven@lerner.co.il&ver=1* ***URL***-t. A válasz, amelyet a kiszolgáló (helytelenül) ***text/html MIME*** típusként azonosít, a következő formátumú sima szöveges állományt tartalmazza:

|A|B|

A figyelemfelhívók a következőképpen értelmezhetik a választ:

- Az A általában a felhasználó feliratkozásában található olvasatlan weblog-bejegyzések számát mutatja.
- Amennyiben a megadott email cím nincs regisztrálva, az A értéke ***-1***.
- Amennyiben ***B*** nem üres, a frissítő lap címére mutat. A dokumentáció nem túl bőbeszédű a frissítő lappal kapcsolatban. Feltételezem ez a lap inkább az embereknek mint programoknak szól, hiszen lehetetlen, de legalábbis nagyon nehéz lenne beazonosítani az összes ***Notifier API***-t használó programot, amelyeknek frissítésre lehet szüksége.

Egy modern magas szintű nyelveben nagyon könnyen fel tudjuk használni a ***Notifier API***-t. Az írás születésekor ***Bloglines*** ügyfélkönyvtárak állnak rendelkezésre ***Perl***, ***Python*** és ***Ruby*** nyelveken. Én a ***Perl*** változatot használom (a ***CPAN***-on ***WebService::Bloglines*** található), de előfordulhat, hogy szívesebben használunk saját verziót, egy másik verziót, esetleg mindkettőt.
Nézzünk egy egyszerű parancssoros programot, amely kiírja, hogy ***„Új blog érkezett!”*** ha a ***Bloglines*** szerint új üzenetek várakoznak, vagy ***„Nincs új blog”*** ha már mindent elolvastunk:

```
#!/usr/bin/perl
use WebService::Bloglines;
my $username = 'reuven@lerner.co.il';
my $password = 'MYPASS';
my $bloglines = WebService::Bloglines->new(
                      username => $username,
                      password => $password);
my $unread_blogs = $bloglines->notify();
if ($unread_blogs)
{
print "'$unread_blogs' új blog üzenete van!\n ";
}
else
{
print "Nincs új blog.\n"
}
```

A `$bloglines->notify()` érték az olvasatlan küldemények számát tartalmazza és a nem az olvasatlan weblogok számát. Amennyiben 15 olvasatlan üzenetünk van 5 különféle weblogban, a `$bloglines->notify()` 15-öt és nem 5-öt ad vissza. Továbbá, a belső Bloglines adatbázis állapotát tükrözi. Azaz, ha a ***Keep New*** dobozra bökünk a weblog bejegyzés végén, ez is bekerül a `$bloglines->notify()` számlálóba. Amennyiben hibás e-mail címet adunk meg, programunk végzetes hibával tér vissza, jelezve, hogy hibás felhasználónevet adtunk meg. Hibás jelszó megadásnak nincsenek következményei a ***Notifier API***-ban, hiszen ez az információ nyilvánosan elérhető.

Blogroll API

A ***Bloglines*** másik eszköze, mint korábban említettük, a ***Blogroll API***. A ***blogroll*** lényegében egy webloglista amit az adott szerző érdekesnek talál és gyakran olvas. Ha érdekesnek találjuk valakinek a weblogját olvasgatni, elképzelhető, hogy kíváncsiak vagyunk olvasmánylistájára is. A ***Bloglines*** esetében a blogroll egyszerűen a felhasználóhoz rendelt feliratkozások listája. Mostanáig azt állítottuk, hogy a ***Bloglines*** felhasználói név megegyezik felhasználó e-mail címével. Ez azonban nem teljesen igaz. Amennyiben a ***Bloglines***-t csak saját magáncéljainkra használjuk, másokkal soha nem osztjuk meg a feliratkozásainkat, az e-mail címünkön kívül tényleg nincs szükségünk másra. Amennyiben azonban közzé szeretnénk tenni feliratkozásainkat, választanunk kell egy felhasználói nevet, amelyre hivatkozni lehet. Például az én regisztrációs e-mail címem ***reuven@lerner.co.il*** míg a felhasználói nevem `reuven`. Ez a megkülönböztetés az első néhány hónapnyi ***Bloglines*** használat során nem igazán volt világos számomra, igaz, úgy tűnik most már valamivel jobban reklámozzák a dolgot.
Amennyiben a felhasználó már szerzett magának egy felhasználói nevet valamint megosztotta feliratkozásait, a következő ***HTML*** és ***JavaScript*** felületen keresztül kérhetjük le a felhasználó blogrollját: ***http://www.bloglines.com/public/reuven***. Ha a blogroll eredményét inkább ***HTML*** formátumban szeretnénk megnézni, azt a következő alakú URL segítségével tehetjük meg: ***http://rpc.bloglines.com/blogroll?id=reuven&html=1***.
Ugyanakkor a webszolgáltatások alapötlete éppen az, hogy az adatokat gépek számára olvashatóvá tegyük és ezáltal a számítógépek tárolhassák és feldolgozhassák. A ***Bloglines*** feliratkozás listáinak exportálásához ***Dave Winer*** 2000-ben készített ***OPML (Outline Processor Markup Language)*** formátumát használja. Ez ugyan nem része a ***Bloglines Webszolgáltatások*** szabályainak, de a következő ***URL***-ről könnyen begyűjthetjük: ***http://www.bloglines.com/export?id=reuven.***
Valamennyi fenti példában a ***Bloglines*** felhasználónevet annak a felhasználónak a nevére kell lecserélnünk akinek a blogjegyzékére kíváncsiak vagyunk. Nem mindenkit teszi nyilvánossá a blogjegyzékét, így aztán letöltéskor könnyen találkozhatunk hibaüzenetekkel. Az ***OPML***-t letöltés után fel kell dolgoznunk valamilyen eszközzel, például a ***CPAN XML::OPML*** moduljával.

Összefoglalás

Mint láthatjuk, a webszolgáltatásokhoz szánt ***Bloglines API*** szabad utat enged a harmadik féltől származó alkalmazásoknak. Egyre nagyobb az esély olyan alkalmazások létrehozására, amelyek ugyan ***HTML***, ***XML*** és ***HTTP*** nyelveket használnak mégsem kötődnek webböngészőhöz. A ***Notifier*** és ***Blogroll API*** megjelenése csak a kezdet. Mint korábban láthattuk a ***Sync API*** lehetővé teszi, hogy a fejlesztők hatékonyan készítsenek új ***GUI*** felületet és alkalmazásokat a ***Bloglines*** által tárolt és begyűjtött tartalomhoz. Következő cikkben belepillantunk a ***Sync API***-ba, és egy egyszerű alkalmazást építünk a ***Bloglines*** alapokra.

Linux Journal 2005. január, 129. szám

Reuven M. Lerner

Paranoid Pingvin – Linuxos VPN Módszerek

Vajon melyik privát hálózat felel meg nekünk? Mick végigszalad a lehetőségeken és bemutatja a győzteseket, valamint kapunk néhány hasznos tanácsot is.

A ***virtuális magánhálózat (virtual private network; VPN)*** igen hasznos és kényelmes dolog. Az úton lévők biztonságosan csatlakoznak vele saját otthoni hálózatukhoz utazás közben; a földrajzilag megoszló társaságok nyilvános sávszélességet használó ***WAN*** kapcsolatként alkalmazzák; a drótnélküli ***LAN*** felhasználók pedig újabb védelmi rétegként használják WLAN kapcsolataik felett.

Linuxra számos ***VPN*** csomag létezik: ***FreeS/WAN***, ***OpenS/WAN***, ***PoPToP***, ***OpenVPN*** és ***tinc***, hogy csak néhányat említsünk. De hogyan választhatjuk ki a megfelelőt a megfelelő feladatra? Nos, ebben a cikkben éppen ezzel foglalkozunk.

VPN szerkezet

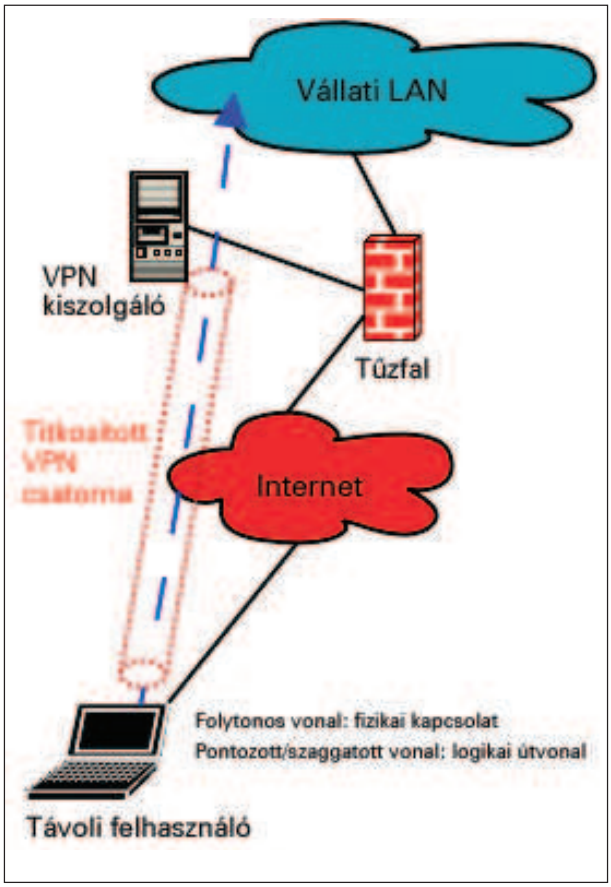
A ***VPN***-ek általában két különféle feladatot látnak el.

Az egyik feladat a felhasználóknak lehetővé tenni az otthoni hálózathoz történő titkosított csatlakozást egy megbízhatatlan médiumon, például az interneten vagy ***wireless LAN (WLAN)*** rendszeren keresztül. Az *1. ábra* a távoli-elérés összeállítást mutatja be.

Az 1. ábrán látható szaggatott kék adatfolyam a teljes üzleti ***LAN*** hálózat elérését jelképezi. A gyakorlatban a távoli elérésű ***VPN*** csatornák ***hozzáférési listák (Access Control List; ACL)*** vagy tűzfalszabályok alapján korlátozhatják ezt az elérést. Az elérést akár egyetlen gép egyetlen alkalmazására is korlátozhatjuk, ahogy az ***SSL-VPN*** rendszerekben ezt általában meg is teszik (az ***SSL-VPN*** rendszerrel hamarosan foglalkozunk).

Az egyszerűség kedvéért az 1. ábra egyetlen ügyfelet mutat be; természetesen az ilyen felállás szinte mindig több ügyfelet tartalmaz. Más szóval, a távoli-elérés megoldás esetében ügyfél-kiszolgáló rendszerben dolgozunk, ahol egyetlen ***VPN*** kiszolgáló vagy gyűjtő távoli felhasználók százaival vagy akár ezreivel létesíthet kapcsolatot. (Ebben a cikkben az ügyfél-kiszolgáló kifejezést kibővített és nem a bizonyos programfejlesztési értelemben használom.)

Bár az 1. ábra a ***VPN*** kiszolgálót az üzleti ***LAN VPN*** végpontjaként mutatja be, tűzfalat is használhatunk erre a célra. Üzleti és ingyenes tűzfalak egyaránt megfelelnek, a ***Linux iptables/Netfilter*** is támogatja a ***VPN*** protokollokat. Fontos megjegyezni, hogy amikor a cikkben csatornát említek, mindig titkosított csatornára gondolok. Igen tudom,



1. ábra Távoli elérésű VPN egy távoli rendszert kapcsol a hálózathoz

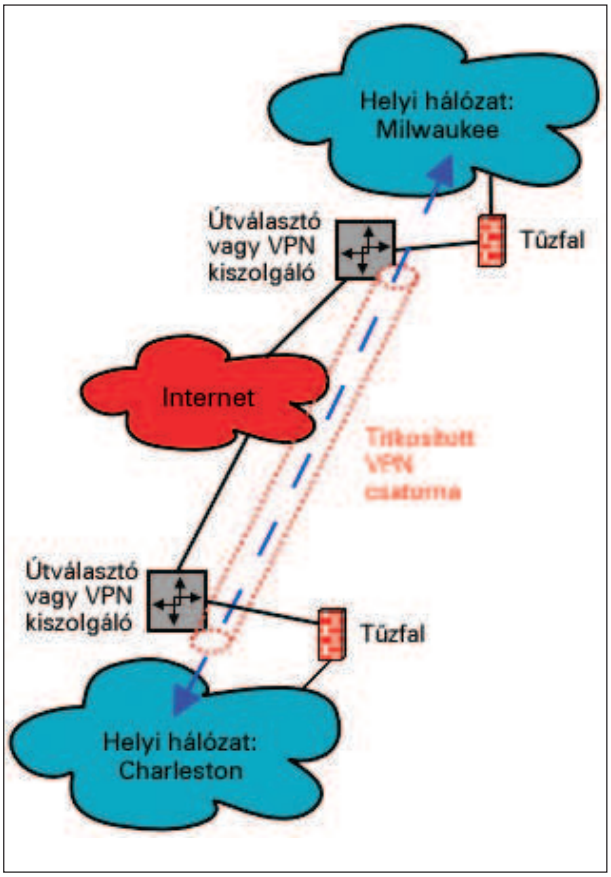
elméletileg a csatorna csak annyit jelent, hogy az egyik adatfolyamot a másikba csomagoljuk. Azonban az egész ***VPN*** lényege a titkosítás, ezért ebben a környezetben a csatorna titkosítást jelent.

A második ***VPN*** alkalmazási lehetőség, amikor két hálózat között titkosított ponttól-pontig kapcsolatot létesítünk, valamilyen nem megbízható médiumon keresztül. Míg a távoli elérésű ***VPN***-ek ügyfél-kiszolgáló modellt alkalmaztak, a ponttól pontig csatornák ***egyenrangú (peer-to-peer)*** szerkezetben dolgoznak. A ***ponttól-pontig típusú VPN szerkezetet*** a *2. ábra* mutatja be.

A ponttól pontig *VPN* felállásban gyakran alkalmazunk útvonalválasztókat. A *Cisco IOS* útvonalválasztó operációs rendszer például több különféle *VPN* protokollt is támogat. A tűzfalak és a dedikált *VPN* gyűjtők/kiszolgálók felhasználhatóak *VPN* végpontként.
A *VPN* szerkezet ezzel a két problémával foglalkozik. Ezen kívül két szerkezeti kérdéssel érdemes még foglalkozni, a ***hálózati címfordítással (Network Address Translation; NAT)*** és a teljesítménnyel.
A legtöbb *VPN* protokoll esetében a *NAT* gondokat okozhat. Ugyanis a *VPN* kiszolgálóknak általában nem lehet lefordított címe. Ez az oka annak, hogy az 1. és 2. ábrában egyik *VPN* végpont sincs az üzleti *LAN* hálózatokban, kivéve a 1. ábrán, hiszen a távoli elérésű ügyfélre mindez nem vonatkozik.
A *NAT* probléma megkerülésének egyik lehetősége, ha a tűzfalat használjuk *VPN* kiszolgálónak, ezzel azonban egy másik kérdés is felmerül: a *VPN* csatornák ugyanis jelentős *CPU* időt emészthetnek fel. Amennyiben a tűzfalunknak nincsen titkosításgyorsító kártyája és nem csak néhány *VPN* csatornát szolgálunk ki, jobb ha külön *VPN* kiszolgálót használunk és nem a tűzfalat alkalmazzuk *VPN*-re.
Az alapokkal megvolnánk, lássuk a *Linux VPN* programjait.

FreeS/WAN és OpenS/WAN

Az *IPSec* protokoll, amely valójában az *Internet Protokol (IP) v6* változatűban felbukkanó biztonsági fejléceinek megvalósítása az *Ipv4* rendszer keretein belül, a legnyitottabb, legerősebb és legbiztonságosabb *VPN* protokoll mind között. Egyben a leggyakoribb is. Az *IPSec* támogatása ma már lényegében az összes számítógépi és hálózati eszköz operációs rendszer része lett. *Linux* alatt a *FreeS/WAN* és *OpenS/WAN* projekteket használhatjuk.
A *FreeS/WAN* rendszerrel korábban részletesen foglalkoztam. Dióhéjban a *FreeS/WAN Linux* rendszerünket kibővíti néhány rendszermag modullal és felhasználói programmal. Abból, hogy az *IP* protokoll a rendszermag része, sejthető, hogy a kiterjesztéseknek is a rendszermagban kell lenniük.
A *Linux 2.6* rendszermag már tartalmazza a *26sec* nevű *IPSec* modult. A *Red Hat Enterprise Linux*-ban fellelhető *Linux 2.4* rendszermagok úgyszintén tartalmazzák, itt ugyanis a *26sec* visszahozott (backported) modulját alkalmazzák. Amennyiben már van *IPSec* rendszermag modulunk, csak a *FreeS/WAN* felhasználói térben futó programjait kell telepítenünk.
A *FreeS/WAN* tetszés szerinti *Linux* terjesztésre telepíthető (az én kedvencem, a *SuSE*, eleve tartalmazza). Ugyanakkor a *FreeS/WAN Projekt* mostanában alakul át, ezért ha a terjesztésünk nem tartalmazza a *FreeS/WAN* rendszert és forrásból kell telepítenünk, jobban járunk ha az *OpenS/WAN* rendszert telepítjük.
Az *OpenS/WAN* projektet a *FreeS/WAN* fejlesztőinek azon csoportja indította útjára, akik nem voltak elégedettek a *FreeS/WAN* projekt alakulásával. Így amikor a *FreeS/WAN* befejeződött, az utódja az *OpenS/WAN* lett. Valószínűleg a nagyobb *Linux* terjesztők hamarosan lecserélik saját *FreeS/WAN* csomagjaikat az *OpenS/WAN* változatra. A legfrissebb *OpenS/WAN* forráskódot az *OpenS/WAN* weblapról tölthetjük le (lásd a hálózati forrásokat).



2. ábra Ponttól-pontig VPN rendszerek két hálózatot kötnek össze

A *FreeS/WAN* és *OpenS/WAN* előnyei:

- ***Kiforrottság***: az egyik legöregebb *Linux VPN* technológia.
- ***Biztonság***: Az *IPSec* ellenálló, hatékony és jól megtervezett protokoll.
- ***Együttműködés***: Más operációs rendszerek valószínűleg rendelkeznek *IPSec* ügyféllel amely együtt tud működni a *Free/OpenS/WAN* rendszerünkkel.
- ***Rugalmasság***: Az *IPSec* egyaránt kiváló ügyfél-kiszolgáló és pontól-pontig *VPN* rendszerekhez.

Hátrányai:

- ***Összetettség***: Az *IPSec* viszonylag nehezen érthető, és digitális hitelesítésre van szüksége.
- ***Erő:*** ha mindössze annyit szeretnénk elérni, hogy a távoli felhasználók az egyik belső rendszerünk alkalmazását elérjék, az *IPSec*-et használni az tipikusan az ágyúval verébre lövöldözés esete. Az *IPSec*-et arra tervezték, hogy egész hálózatokat kapcsoljon össze.

Összefoglalva, ha a cikk elolvasása után még mindig nem tudjuk biztosan melyik *VPN* megoldást kellene választanunk, javaslom alapesetben maradjunk a *FreeS/WAN* vagy *OpenS/WAN* megoldásnál. Az *IPSec* messze a legkiforrottabb és legbiztonságosabb *VPN* technológia *Linux* alatt. Véleményem szerint ezek az előnyök bőségesen ellensúlyozzák az összetettség miatti hátrányt. A csomagok beállításával és használatával kapcsolatos további információkat a *FreeS/WAN* és *OpenS/WAN* weblapjain találunk.

OpenSSH

Az ember hajlamos azt hinni, hogy az *OpenSSH* egyszerűen csak egy távoli bejelentkezési eszköz. Csakhogy az *SSH* protokoll nem csak a héj (shell), hanem bármilyen, egyetlen *TCP*-kaput használó szolgáltatás átvitelét támogatja, mégpedig biztonságos csatornán, a `-L` és `-R` kapcsolók segítségével.
Például, tegyük fel van egy biztonságos héj (secure shell) kiszolgálónk a tűzfal mögötti de nyilvánosan elérhető *DMZ* hálózatunkon, valamint egy *Microsoft SQL* kiszolgálónk a belső hálózaton. Ha készítek egy tűzfal szabályt, amely a *MS-SQL* műveleteket engedélyezi az *SSH* kiszolgáló és a *MS-SQL* kiszolgáló között és az *SSH* kiszolgálóm engedélyezi a kaputovábbítást, létrehozhatok egy *SSH* csatornát egy tetszőleges távoli gép és az *SSH* kiszolgálóm között amelyen keresztül a távoli adatbázis ügyfelek lekérdezései először az *SSH* kiszolgálóhoz kerülnek, majd onnan a *MS-SQL* kiszolgálóhoz továbbítódnak. A távoli gépemen kiadott *SSH* parancs a következőképpen nézne ki:

```
bash-#> ssh -L 11433:ms-sql.server.name:1433
➥myaccount@remote.ssh-server.name
```

ahol az `ms-sql.server.name` a *MS-SQL* kiszolgáló neve vagy *IP* száma, a `remote.ssh-server.name` pedig *DMZ*-sített *SSH* kiszolgáló neve vagy *IP* száma.
Még *PPP*-t is küldhetünk *SSH*-n keresztül, ami lényegében azonos feladatot lát el, mint az *IPSec,* azaz a két hálózat között minden adatot továbbit. Mindazonáltal ez az egyik legkevésbé hatékony megoldás; Sokkal több adminisztrációt igényel mint a cikkünkben bemutatott egyéb eszközök és módszerek.
Összefoglalva, az *OpenSSH* leginkább egy adott gépen futó, adott alkalmazás adatainak átküldésére használható; ilyen felállásban távoli-elérésre és ponttól pontig *VPN* megoldásokban is használható. Kevésbé használható ugyanakkor a távoli hálózatok vagy felhasználók összes adatának továbbítására.
Az *ssh* és *sshd_config* kézikönyvoldalakon további információkat találunk az *OpenSSH* kaputovábbítási képességeiről.

Stunnel

Az *Stunnel* nevű *SSL* csomagoló lényegében az *SSH* kaputovábbítással azonos képességeket nyújt. A legtöbb mai *Linux* terjesztésben alapcsomag.
Az *Stunnel* és az *SSH* közötti főbb különbségek, hogy az *Stunnel* sokkal korlátozottabb; kizárólag titkosított kaputovábbításra lehet használni. Ezen felül, mivel az *Stunnel* tulajdonképpen valamiféle előlap az *OpenSSL*-hez, az *Stunnel* használatához digitális bizonyítványokat kell telepítenünk, ami elég sokat ront az egyszerűségén. Egyéb tekintetben *VPN* eszközként az *Stunnel* az *OpenSSH*-hoz hasonló korlátozásokkal használható.
Az *Stunnel* beállításával és használatával kapcsolatos további információkat az *stunnel* kézikönyv oldalon, a *Stunnel* weblapon, no és a *„Kódolatlan szövegekkel dolgozó alkalmazások feltámasztása az Stunnel segítségével" (Linuxvilág 2004. október)* című korábbi cikkemben találhatunk.

OpenVPN

Az *OpenVPN* egy *SSL*/*TLS*-alapú felhasználói *VPN* eszköz, amely becsomagolja az összes forgalmat hagyományos *UDP* és *TCP* csomagokba két *VPN* végpont között (hagyományos szó itt olyan értelemben értendő, hogy a rendszermag *IP* veremben nincs szükség változtatásokra).
Az *OpenVPN* azért jött létre, mert a szerzője *James Yonan* szerint a világnak az *IPSec*-nél egyszerűbb megoldásra is szüksége van.
Minthogy semmilyen speciális rendszermag módosításra nincs szükségünk, az *OpenVPN* teljes egészében a felhasználói térben fut, így sokkal könnyebb az operációs rendszerek közötti átvitele mint az *IPSec* megoldásoké. Ezen kívül az *OpenSSL* könyvtárak alkalmazásával, az *OpenVPN*, akárcsak az *Stunnel*, a lehető legkevesebb felesleges újrafelfedezéssel oldja meg a problémát. A saját készítésű titkosítórendszerekkel szemben (ilyen a *CIPE*, *tinc* és *VPN* csomag, lásd alább), az *OpenVPN* valamennyi kritikus műveletét az *OpenSSL* végzi. Persze maga az *OpenSSL* sem hibátlan, de a biztonsági hiányosságok tekintetében folyamatosan a figyelem középpontjában áll és az *Open Source* közösség legkiválóbb titkosítási programozói tartják karban.
Az *OpenVPN* jó választás pontól-pontig *VPN* készítéshez, de a 2.0-ás verzió előtt (amely 2004 novemberében még béta változatban volt csak elérhető), az *OpenVPN* korlátozott képességekkel rendelkezik, ugyanis csak egyetlen csatornát tud átvinni egy adott kapunk. Amennyiben az *OpenVPN*-t távoli-elérésű *VPN* csatornaként szeretnénk használni tíz különféle felhasználóhoz, tíz külön *OpenVPN* fogadót kell indítanunk, mindegyiket saját *UDP* kapuval. Tehát fel kell használnunk az *UDP 10201*, *UDP 10202* és *UDP 10203* valamint még hét további kaput. Ezért ha a *OpenVPN*-t tényleg távoli *VPN* elérésre szeretnénk alkalmazni és nem csak egy két felhasználónk van, sokkal jobban járunk az *OpenVPN 2.0* változattal (még ha béta állapotú is).
Az *OpenVPN* beépítve megtalálható a *SuSE Linux 9.1* rendszerben és valószínűleg egyéb terjesztésekben is.
Az *OpenVPN* weblapján megtaláljuk a beállítási információkat és a legfrissebb *OpenVPN* programot.

PoPToP and the Linux PPTP Client

Az *IPSec* nem az egyetlen alacsony szintű *VPN* protokoll az Interneten. A *Microsoft Point-to-Point Tunneling Protocol (PPTP)* megoldásnak is vannak követői, leginkább azért mert ez a rendszer vált a *Microsoft* kiszolgáló operációs rendszer szabványává a *Windows NT 4.0* óta, valamint mivel az *IPSec*-el ellentétben amely csak *IP* csomagokon keresztül tud csatornázni, a *PPTP* nem csak *IP* csomagokon de más protokollokon például *NETBEUI* vagy *IPX*/*SPX* rendszeren keresztül is működik.
A *Linux* alatt két *PPTP* változat létezik, a kiszolgáló oldali *PoPToP* és az ügyfél oldali *Linux PPTP*.
Bár ha nem *IP* protokollokon szeretnénk csatornázni igen hasznos lehet és általános minden *Windows* kiszolgáló környékén megtalálható, a *PPTP*-nek van egy nagy hátránya. Amikor *Bruce Schneier* és *Dr Mudge* megvizsgálták a *Windows NT 4. PPTP* megoldását 1998-ban, komoly biztonsági hiányosságokat fedeztek fel, amelyeket csak részben orvosolt a nem sokkal később kiadott *MSCHAPv2* javítás.

Az *MSCHAP* azonosítási protokoll, amelyre a *PPTP* épül, *Schneier* és *Mudge* által talált legrosszabb biztonsági hibák forrásának bizonyult. *Schneier* honlapján megtekinthetjük a vizsgálatuk eredményét (lásd a forrásokat).
Schneier és *Mudge* a *Windows NT 4.0* rendszert vizsgálták; de mi a helyzet a *Linux PoPToP* kiszolgálóval? A *PoPToP* weblap szerint (a *„PoPToP Kérdések és Válaszok”* fejezetben): *„A PoPToP ugyanolyan biztonsági problémákkal küszködik, mint az NT kiszolgáló (ez azért van mert Windows ügyfelekkel dolgozik).”*
Nem javaslom a *PPTP* használatát, hacsak nem tudjuk a *PPTP* kiszolgálónkat és valamennyi *PPTP* ügyfelünket rábírni az *MSCHAPv2* használatára (sajnos nem minden *Windows* változat támogatja a *MSCHAPv2*-t) és akkor is csak abban az esetben ha valami olyasmit csinálunk amit egyszerűen nem lehet megoldani *IPSec*-el. Az *IPSec* sokkal jobban megtervezett és bizonyítottan biztonságosabb. Ezen kívül a nem *IP* alapú hálózati protokollok ma már nem olyan létfontosságúak mint valaha; a *Windows* és a *Novell Netware* bármit meg tud csinálni *IP*-n keresztül.
Senki ne értse félre, nem állítok olyasmit, hogy „ne használd a PPTP-t mert béna”. Mint előző hónapban kifejtettem, a biztonság , a kockázatelemzésről szól, és nem valamiféle utópisztikus, tökéletes biztonság kereséséről. Miután elolvastuk *Schneier* és *Mudge* vitáját, a *Microsoft* válaszát és a *MSCHAPv2*-t, majd gondosan megvizsgáltuk a vállalatunk igényeit és képességeit, elképzelhető, hogy úgy döntünk, hogy a *PPTP* elfogadható kompromisszumot jelent a biztonság és a funkciók terén – csak aztán senki ne mondja hogy ***én*** javasoltam!

Egyéb Linux VPN csomagok

Három további *Linux VPN* eszközt érdemes még megemlíteni itt, hiszen néha láthatunk rájuk hivatkozásokat. Kettő használatát nem szívesen javaslom, a harmadikban nem vagyok biztos.
A *CIPE* és a *vtun* lényegét tekintve azonos az *OpenVPN* rendszerrel. A forgalmat *UDP* vagy *TCP* csomagokba zárják. Az *OpenVPN*-el ellentétben azonban házilag fejlesztett titkosítási rendszereket használnak az *OpenSSL* helyett. Pontosabban olyan hagyományos titkosítási eljárásokat használnak mint a *Blowfish* vagy az *MD5*, de saját megvalósítással (folyamat-kulcs készítés, felhasználó azonosítás és egyebek). Minthogy a titkosítás programozásban éppen a megvalósítás a legnehezebb rész, igen veszélyes lehet, és lám, *Peter Gutmann* titkosítási szakember súlyos biztonsági hibákat talált a *CIPE* és *vtun* rendszerekben.
Amennyire tudom, egyik esetben sem javították *Gutmann* azonosította hibákat. Ráadásul úgy tűnik sem a *CIPE* se a *vtun* rendszert nem fejlesztik már aktívan (a *CIPE*-t biztosan nem), ami önmagában éppen elég indok, hogy messziről elkerüljünk egy biztonsági alkalmazást, kivéve, ha egy *Linux* terjesztés része, ahol a csomag karbantartói maguk készítenek foltokat. Ezekből az okokból kifolyólag nem javaslom se a *CIPE* se a *vtun* használatát.
A *tinc*, akárcsak a *CIPE* és a vtun, saját kódolási megoldást használ a *VPN* forgalom titkosított *UDP* csomagokba zárására. És akárcsak a fenti csomagokban, *Gutmann* a *tinc*-ben is talált hibákat, a korábban említett vizsgálata során. A *tinc* fejlesztői azonban a *CIPE* és *vtun* csapataival ellentétben hitelt érdemlő módon válaszoltak Gutmann felfedezéseire; legalábbis az én szemszögemből nézve. Ami az illeti, részemről IANAC! (IANAC = „I Am Not A Cryptographer” ; „Én Nem Vagyok Kriptográfus”.) Úgy tűnik, van valami fogalmuk róla mit is csinálnak.
Önökre bízom a *tinc* weblapjának felderítését, és *Gutmann* lapjának elolvasását (amely komoly kutatási jelentés lévén elég súlyos darab), no és pár *Google* keresés elvégzését *Gutmann* megállapításainak sorsát illetően. Ezek alapján mindenki eldöntheti, hogy a *tinc* éppen az amire szüksége van, vagy inkább nemkívánatos biztonsági kockázatot jelentene a könnyen elérhető *OpenS/WAN* és *OpenVPN* helyett.

SSL-VPN

Végül, ejtsünk néhány szót a számos üzleti *VPN* termékben megjelenő népszerű új megközelítésről: az *SSL-VPN* rendszerről. Az *SSL-VPN* gyakorlatilag pontosan úgy működik mint az *Stunnel* és az *SSH* kaputovábbítás. A hálózati tranzakciókat szolgáltatás- és kiszolgáló-alapon viszi át, és nem felső kapcsolati szinten. A többi megközelítéssel szemben az *SSL-VPN* termékek a végfelhasználónak központosított webes felületet nyújtanak ahol a *VPN* rendszerben kezelt összes kiszolgáló/szolgáltatás hivatkozásként megjelenik. Amikor a felhasználó rákattint egy hivatkozásra, általában egy *Java* kisalkalmazás töltődik le, amely az alkalmazás ügyfélprogramjaként működik.
Valamennyi *SSL-VPN* kiszolgáló termék amellyel találkoztam, üzleti megoldás volt, de mivel az ügyféloldal általában *Javaban* íródott és rendszerfüggetlen, a *Linux* rendszerek is dolgozhatnak *SSL-VPN* ügyfelekként.

Összefoglalás

A *FreeS/WAN* és *OpenS/WAN* (lehetőség szerint az utóbbi) és az *IPSec* nyújtják valószínűleg a legbiztonságosabb és leghatékonyabb *VPN* megoldást *Linuxos* eszközökön.
Az *OpenVPN* egyszerűbb, ugyanakkor kevésbé alaposan vizsgált alternatívát jelenthet. Az *OpenSSH* és az *Stunnel* jó csomagoló megoldást jelenthetnek amikor az előzőek használata túlzás lenne. Más *Linux VPN* eszközök is elérhetőek, de egyesek bizonyítottan veszélyesek, mások esetében pedig a zsűri még nem döntött. Melyik *VPN* eszköz lesz a legmegfelelőbb számunkra? Természetesen ezt nem tudom megmondani anélkül, hogy az adott igényeket és erőforrásokat ismerném. De remélem ez a kis összefoglaló legalább segít elindulni.

Linux Journal 2005. február, 130. szám

A cikk forrásai: ➲ ***www.linuxjournal.com/article/7923***

Mick Bauer (mick@visi.com)
Biztonsági szakember, a Linux Journal biztonsági témákkal foglalkozó szerkesztője, biztonsági tanácsadó a Minnesota állambeli Minneapolisban található Upstream Solutions LLC Inc.-nél.

Biztonságos FTP-szolgáltatás üzemeltetése vsftpd használatával

Több webhostinggal foglalkozó rendszergazda egyik nagy dilemmája, hogy hogyan lehetne biztonságos, ám mégis felhasználóbarát fájlelérést biztosítani.

Így vagy úgy, de előbb-utóbb minden rendszergazda szembekerül azzal a ténnyel, hogy a sikeres szerverüzemeltetéshez nem elég, ha rendszerét szigorú biztonsági megfontolásokkal őrzi. Gyakran előfordul, hogy egyébként rajta kívül senki nem tudja kényelmesen igénybe venni a szolgáltatásokat. A webhosting egy tipikus példája annak, amikor viszonylag nagy ügyfélkörnek kellene biztosítani azt, hogy hozzáférjen a (saját) weblapjához. Feltölthessen, letölthessen; magyarul módosítani tudja annak tartalmát. Ráadásul mindezt a lehető legegyszerűbben.
Az ilyen helyzetekben egyre gyakoribb a webes fájlkezelő, vagy az ***SCP/SFTP*** bevezetése. Már-már meg is bontaná a sörét a rendszergazda (kényelmesen hátradőlve székében), ám a következő pillanatban csörög a telefon, hogy nem lehetne-e inkább mégis ***FTP***-t használni. Majd az első elégedetlenkedők után jönnek a többiek is… Ugye ismerős a helyzet?
A közhiedelem úgy tartja, hogy az ***FTP*** használata nem biztonságos. Nos, sok esetben valóban helytálló lehet ez a megállapítás, de azért korántsem igaz. Jelen cikkben azt próbálom meg bebizonyítani, hogy hogyan lehet ezt a sokak által hangoztatott feltételezést megcáfolni. Mégpedig azzal, hogy létrehozunk egy olyan FTP kiszolgálót, amelyben egyesül a biztonság, az egyszerűség, a gyorsaság, és nem utolsó sorban alkalmas arra, hogy felhasználóinknak barátságos hozzáférést biztosítsunk személyes fájljaikhoz.
Mick Bauertől, a ***Linux Journal*** biztonsági témákkal foglalkozó szerkesztőjétől remek cikket olvashattunk a ***Linux-világ*** múlt év szeptemberi számában arról, hogy miképp lehet a ***vsftpd***-vel egy igazán remek ***anonymous FTP*** szervert készíteni. Ezt a gondolatmenetet követve, mi most elkészítünk egy ***PAM*** segítségével ***MySQL*** adatbázisból hitelesítő, opcionális ***SSL*** támogatással felvértezett ***vsftpd***-t; mellyel így élvezhetjük a több szempontból is hasznos virtuális felhasználó-kezelés előnyeit (szemben az ***SCP/SFTP*** megoldással, ahol kényelmetlenebb lenne ezt kivitelezni).
Hogy miért is kellenek ezek? A ***MySQL*** alapú háttérre azért van szükség, mert nagyobb számú felhasználói fióknál jobb teljesítményt nyújt, mint ha például egy ***Berkley DB*** adatbázisból hitelesítenénk; továbbá adminisztrálni is jóval egyszerűbb. Az ***SSL*** segítségével pedig biztosítani fogjuk a felhasználó és a szerver közötti titkosított adatáramlást, ezáltal megvédve a klienst például az illetéktelen lehallgatástól. Segítségével a jelszavak (és az adat is, ha úgy akarjuk) nem ***nyílt szöveg (plain-text)*** formátumban fognak mozogni a hálózaton, mint ahogyan azt egyébként tennék.

Vágjunk is bele!

Szerencsémre ***Mick*** már ismertette a ***vsftpd*** telepítését többféle disztribúció szemszögéből is, így ha minden jól ment, akkor jelenleg alapbeállításokkal ott figyel számítógépünkön a szoftver. Állítsuk le, és tegyük biztos helyre a mostani konfigurációs fájlt:

```
debian:~# mv /etc/vsftpd.conf /dev/null
```

Mi most teljesen a nulláról fogjuk újra felépíteni, megtárgyalva minden egyes opcióját.

A munkálatokhoz szükséges hozzávalók telepítése

Mielőtt belefognánk bármibe is, telepítsük a megfelelő szoftvereket. Szükségünk van tehát egy ***mysql*** szerverre (ha még nincs) a felhasználói nevek és jelszavak tárolásához, egy ***openssl*** csomagra az ***SSL*** támogatás biztosításához, valamint a ***libpam-mysql*** modulra a ***PAM*** és ***MySQL*** összekapcsolásához. ***Debian Sarge*** alatt ezeket nagyon egyszerűen össze is tudjuk szedni.

```
debian:~# apt-get install mysql-server openssl
➥libpam-mysql
```

Igazság szerint, akik már ismerik a ***MySQL***-t, és van egy megszokott kezelőfelületük az adminisztrálására, azok természetesen könnyedén fogják majd kezelni a felhasználókat. Akiknek viszont nincs, azoknak nem érdemes csak ezért telepíteniük egyet, hiszen a cikk végére látni fogjuk, hogy sokkal jobban járunk, ha megírjuk a saját szkriptünket az adminisztrációs munka megkönnyítésére.

A MySQL alapvető biztonsági beállításai

Ha most telepítettünk életünkben először ***mysql*** kiszolgálót, és még nem olvastuk el a dokumentációkat róla, akkor – bár a cikk alapvetően nem erről szól – leírom, hogyan tegyük meg a minimális biztonsági óvintézkedéseket.

```
debian:~# mysql -u root
mysql> USE mysql
mysql> DELETE FROM user WHERE User="";
mysql> UPDATE user SET Password=PASSWORD('ide
➥irjuk a jelszot') WHERE User='root'
mysql> FLUSH PRIVILEGES;
mysql> quit
```

Ez a néhány sor annyit csinál, hogy egyrészt kitörli az anonymous hozzáférést, másrészt megad egy jelszót a root felhasználó(k)nak, mivel telepítés után az üres. Sajnos elég gyakori, hogy ezt a figyelmetlen rendszergazdák így is hagyják. Ilyen esetben viszont sajnos hiába is dolgoznánk azon, hogy biztonságos FTP kiszolgálót hozzunk létre…

Az adatbázis elkészítése

A feladat tehát a következő: hozzunk létre egy `vsftpd` nevű adatbázist, egy `felhasznalok` nevű táblával, amiben a következő mezőkre lesz szükségünk: `nev`, `jelszo`. Természetesen más neveket is használhatunk, amennyiben ezek nem tetszenek.

```
debian:~# mysql -u root -p # bejelentkezünk
➥a MySQL szerverünkre root-ként,
➥immár jelszó használatával
mysql> CREATE DATABASE vsftpd; # elkészítjük
➥a vsftpd adatbázist
mysql> USE vsftpd; # majd ki is jelöljük, mivel
➥ezzel fogunk most dolgozni
mysql> CREATE TABLE felhasznalok ( nev char(16)
➥binary, jelszo char(16) binary ); # létrehozzuk
➥a táblát és a mezőket
mysql> INSERT INTO felhasznalok ( nev, jelszo )
➥values ( 'kisspista', password( 'a jelszo' ));
➥# felvesszük az első felhasználónkat
➥az adatbázisba
mysql> GRANT SELECT, INSERT, UPDATE, DELETE,
➥CREATE, DROP, INDEX, ALTER ON `vsftpd`.* TO
➥'vsftpd_mysql'@'localhost' IDENTIFIED BY
➥'a jelszavunk'; # egy vsftpd_mysql nevű
➥felhasználónak megadjuk a megfelelő jogokat
➥az adatbázisra (és létre is hozzuk egyúttal)
mysql> quit
```

Ezzel fel is készítettük rendszerünket arra, hogy a *vsftpd* gond nélkül információkhoz jusson az adatbázisból a továbbiakban.

A PAM beállítása MySQL hitelesítéshez

A *PAM (Pluggable Authentication Modules)* egy olyan függvénygyűjtemény, amelyben az a legnagyszerűbb, hogy egységes felületet biztosít a különböző programoknak az azonosítási procedúra elvégzésére. A folyamatot végző szoftver csak a *PAM*-mal áll kapcsolatban, a *PAM* pedig onnan veszi az információkat, ahonnan csak akarjuk (jelen esetben például egy *MySQL* adatbázisból). *Linuxunk* eme remek szolgáltatását használjuk például akkor is, amikor *SSH*-val bejelentkezünk szerverünkre. Ilyenkor az *SSH* démon elküldi a *PAM*-nak az általunk megadott felhasználónevet és jelszót, ami (többek közt) megvizsgálja a */etc/shadow* fájl alapján, hogy helyes adatokat adtunk-e meg. Amennyiben igen, visszajelez az *SSH* kiszolgálónak, hogy minden rendben, bejelentkezhetünk. Ma már a szolgáltatások legtöbbje a *PAM*-ot használja hitelesítésre.
Ha belépünk a */etc/pam.d* könyvtárba (bizonyos disztribúciók a */etc/pam.conf* fájlt használják), akkor látni fogjuk, hogy minden egyes szolgáltatáshoz más-más beállítófájl tartozik. A *vsftpd* például a `vsftpd` nevű fájlt hozza létre alapból (régebbi változatok esetén előfordulhat, hogy mást). Tegyük ezt is biztos helyre úgy, ahogyan a cikk elején az alapértelmezett *vsftpd.conf* fájlt. Egy teljesen újat fogunk írni. Adjuk ki a következő parancsokat:

```
debian:~# echo "auth required pam_mysql.so
➥user=vsftpd_mysql passwd=jelszavunk
➥host=localhost db=vsftpd table=felhasznalok
➥usercolumn=nev passwdcolumn=jelszo crypt=2"
➥> /etc/pam.d/vsftpd
debian:~# echo "account required pam_mysql.so
➥user= vsftpd_mysql passwd=jelszavunk
➥host=localhost db=vsftpd table=felhasznalok
➥usercolumn=nev passwdcolumn=jelszo crypt=2"
➥>> /etc/pam.d/vsftpd
```

Ezzel megadtuk a *PAM*-nak, hogy a *pam_mysql* modult használja az azonosításhoz. Természetesen a paraméterezés nagyon lényeges, így ügyeljünk arra, hogy helyesen adjuk meg például azt a jelszót, amit az adatbázis elkészítésekor létrehoztunk annak a *MySQL* felhasználónak (*vsftpd_mysql*), aki majd hozzáférhet az adatbázishoz.

Az SSL kulcs legyártása

Ahhoz, hogy *FTP* szerverünkön biztosítsuk az *SSL* szolgáltatást, szükségünk lesz egy *.pem* formátumú *RSA* kulcsra. Ezt a következőképpen tudjuk legenerálni:

```
debian:~# openssl req -x509 -nodes -newkey
rsa:1024 -keyout vsftpd.pem -out vsftpd.pem
```

Miután válaszoltunk az ez után feltett kérdésekre, létre is jön a `vsftpd.pem` fájl ott, ahol kiadtuk a parancsot. Másoljuk át a */usr/share* mappába, és állítsuk be a megfelelő jogosultságokat rajta.

```
debian:~# cp vsftpd.pem /usr/share
debian:~# chmod 400 /usr/share/vsftpd.pem
```

Problémák az SSL használatával

Az *SSL* használata nagyszerű dolog, azonban van egy hátulütője a dolognak. Sajnos nem minden *FTP* kliens támogatja. Így el kell döntenünk, hogy kötelezővé tesszük-e a felhasználóknak a használatát, vagy sem. Amennyiben maximális biztonságra törekszünk, akkor mindenképp kötelezzük el magunkat (és ügyfeleinket) mellette. Ennyi áldozatot bőven megér, hogy biztonságban tudhatjuk a kommunikációt is a hálózaton.

A vsftpd beállításának előkészületei

Végre elérkeztünk oda, hogy bekonfigurálhatjuk a *vsftpd*-t. No, de mit is akarunk pontosan csinálni? Mik a követelmények a beállításokkal szemben?

A vsftpd.conf

```
# Beállítjuk, hogy daemon-ként induljon el
# a vsftpd
listen=YES
# Tiltjuk a névtelen bejelentkezést
anonymous_enable=NO
# Viszont engedélyezzük a "nem-névtelen"
# felhasználókat (akiket a PAM hitelesít nekünk)
local_enable=YES
# Engedélyezzük a vendég felhasználót.
# Ez nagyon fontos, mivel ez aktiválja
# a virtuális felhasználó-kezelést.
guest_enable=YES
# Beállítjuk, hogy minden bejelentkező felhasználót
# a vsftpd_user felhasználóra mappeljen a szerver
guest_username=vsftpd_user
# Megadjuk a felhasználók egyéni beállításait
# tartalmazó mappa helyét a fájlrendszeren
user_config_dir=/usr/share/vsftpd/users
# Beállítjuk, hogy a vsftpd nevű PAM beállítófájlt
# használjuk az azonosítási procedúrához
pam_service_name=vsftpd
# Engedélyezzük az írást. feltöltést, könyvtár
# létrehozását
write_enable=YES
anon_upload_enable=YES
anon_mkdir_write_enable=YES
anon_other_write_enable=YES
# Beállítjuk, hogy mindenki csak a saját
# könyvtárján belül mozoghasson
chroot_local_user=YES
# Itt adhatjuk meg az ftp root könyvtárat,
# de mi most elővigyázatossági okokból az
# /usr/share/vsftpd/empty -t adjuk meg, később
# rájövünk, hogy miért…
local_root=/usr/share/vsftpd/empty
# Engedélyezzük az SSL támogatást
ssl_enable=YES
# Meghatározzuk a .pem fájl helyét
rsa_cert_file=/usr/share/vsftpd.pem
# Kényszerítjük-e a felhasználókat, hogy
# a bejelentkezési információkat SSL-el titkosítsák?
force_local_logins_ssl=NO
# Kényszerítjük-e a felhasználókat, hogy
# az adatfolyamatot SSL-el titkosítsák?
force_local_data_ssl=NO
# Beállítjuk a megfelelő jogosultságokat
# a feltöltött fájlokra..
anon_umask=022
# Ha akarunk megadni globális sávszélesség
# korlátot, akkor itt megtehetjük. Ha 0-án hagyjuk,
# akkor korlátlanra állítjük. Az értéket byte / sec
# formában értelmezi a program.
anon_max_rate=0
```

Egyrészt ne engedélyezzünk semmilyen anonymous kapcsolódást, csakis azok csatlakozhassanak, akik a *MySQL* adatbázisunkban szerepelnek. Szerepeljen benne az *SSL* lehetősége (jelen esetben nem kényszerítve), a virtuális felhasználó-kezelés, a különböző felhasználók egyéni beállításainak engedélyezése, valamint az, hogy mindenki csak a neki beállított könyvtárban mozoghasson. Érdemes külön figyelmet szentelni a felhasználók egyéni beállításainak lehetőségére. Ez egy remek szolgáltatás, ami a *vsftpd 1.1.0*-s verziójában jelent meg először. Segítségével minden egyes felhasználónévhez más-más beállításokat rendelhetünk, kezdve a sávszélesség korlátozástól, a *home* könyvtár helyéig.

Előkészületként szükségünk lesz egy vsftpd_user nevű felhasználóra a rendszeren, egy /usr/share/vsftpd/empty, egy /usr/share/vsftpd/users, valamint egy ftp root könyvtárra (ami legyen most például a /var/www).

```
debian:~# mkdir -p /usr/share/vsftpd/empty &&
➥chmod 500 /usr/share/vsftpd/empty
debian:~# mkdir /usr/share/vsftpd/users
debian:~# useradd -d /usr/share/vsftpd/empty
➥vsftpd_user
debian:~# passwd vsftpd_user
```

A felhasználók egyéni beállításai

Emlékezzünk csak vissza, az adatbázis létrehozásánál készítettünk egy `kisspista` nevű felhasználót. Hogyan hozzunk neki létre egyedi beállításokat, mint például, hogy hol legyen a *home* könyvtára?

A működési elv nagyon egyszerű. A konfigurációs fájlban megadott helyen (*/usr/share/vsftpd/users*) létre kell hozni egy fájlt a felhasználó nevével megegyezően, majd szépen beleírni azokat a beállításokat, amiknél azt szeretnénk, hogy eltérjen az alapértelmezett értéktől. Hozzuk is létre neki!

```
debian:~# echo "local_root=/var/www/kisspista.hu"
➥> /usr/share/vsftpd/users/kisspista
debian:~# echo "anon_max_rate=20000"
➥>> /usr/share/vsftpd/users/kisspista
```

Ezek után készítsük el *Kiss Pista* tetszetős weboldalát, hogy ne tátongjon ott üresen az a könyvtár. Ne felejtsük el megfelelően beállítani a könyvtár tulajdonosi jogát!

```
debian:~# mkdir -p /var/www/kisspista.hu
debian:~# chown vsftpd_user /var/www/kisspista.hu
debian:~# echo "<HTML><HEAD><TITLE>Kiss Pista
➥weboldala</TITLE></HEAD><BODY BGCOLOR="green"
➥TEXT="red"><CENTER><H1>Felújítás
➥alatt…</H1></CENTER></BODY></HTML>"
➥> /var/www/kisspista.hu/index.html
```

Indítás, bejelentkezés

Ezzel el is készültünk minden beállítással, elindíthatjuk a *vsftpd*-t. Tapasztalataim szerint nem nagyon reagál a konfigurációs hibákra, ha valami nem tetszik neki, akkor egyszerűen nem indul el. Ha viszont semmi probléma nem adódott, akkor itt az idő letesztelni, hogy működik-e a hitelesítés. Valami hasonlót kell látnunk:

```
debian:~# ftp localhost
Connected to localhost.localdomain.
220 (vsFTPd 2.0.1)
Name (localhost:rusty): kisspista
331 Please specify the password.
```

1. lista

Titkosítatlan bejelentkezés

```
04:04:32.660997 IP (tos 0x0, ttl 122, id 36801,
➥offset 0, flags [DF], length: 56)
➥3e44b710.adsl.enternet.hu.4480 >
➥programfiles.hu.ftp: P [tcp sum ok] 1:17(16)
➥ack 21 win 65515
E..8..@.z..&>D...F2=....B..Z-.WpP...q...USER
➥kisspista
04:04:32.687190 IP (tos 0x0, ttl 122, id 36807,
➥offset 0, flags [DF], length: 56)
➥3e44b710.adsl.enternet.hu.4480 >
➥valami.hu.ftp: P [tcp sum ok] 17:33(16)
➥ack 55 win 65481
E..8..@.z.. >D...F2=....B..j-.W.P....PASS pisti9234
```

Titkosított bejelentkezés

```
04:09:27.227510 IP (tos 0x0, ttl  64, id 50447,
➥offset 0, flags [DF], length: 1053) valami.hu.ftp
➥> 3e44b710.adsl.enternet.hu.4509: P 52:1065(1013)
➥ack 143 win 6432
E.....@.@....F2=>D.....@...g9.@P.. `T......J...F..A.
➥.....wq.].....gq...i........
04:09:27.286248 IP (tos 0x0, ttl 122, id 37542,
➥offset 0, flags [DF], length: 230)
➥3e44b710.adsl.enternet.hu.4509 > valami.hu.ftp:
➥P 143:333(190) ack 1065 win 64471
E.....@.z...>D...F2=....g9.@@...P..................
➥vE.I....f`...TqSxK.z...^d....R.
```

```
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0    0     122 Jan 18 05:17 index.html
226 Directory send OK.
```

Ha nem sikerül bejelentkeznünk, akkor nézzük át ismét minden konfigurációs fájlunkat, valamint olvassunk bele a */var/log/mysql/mysql.log* és a */var/log/syslog* naplófájlokba.

Az SSL támogatás tesztelése

Az *SSL* hasznosságára valószínűleg akkor fogunk rájönni, ha a saját szemünkkel látjuk a különbséget a titkosítatlan, valamint a titkosított bejelentkezés között.
Tegyük is ezt meg. A megfigyeléshez szükségünk lesz a `tcpdump` nevű programra, amit ***Debian Sarge*** alatt egyszerűen az `apt-get install tcpdump` paranccsal tudunk beszerezni. Ha megvan, adjuk ki a következő utasítást:

```
debian:~# tcpdump -A host ipcím and port 21 -v
```

A parancsba természetesen helyettesítsük be az `ipcím` helyére azt az IP címet, ahonnan csatlakozni fogunk az *FTP* szerverre a tesztelés ideje alatt. Figyeljük meg a program kimenetét akkor, amikor egy *SSL*-t nem támogató kliens programmal, valamint, amikor egy azt támogatóval jelentkezünk be. A különbség szemmel látható (*1. lista*)
A probléma persze nem ott kezdődik, hogy a titkosítatlan bejelentkezésnél az *FTP* szerverünkön ki tudjuk szűrni a hálózati forgalomból a felhasználói nevet és jelszót, hanem ott, hogy a helyi hálózaton – ahonnét a kliens csatlakozik – kis túlzással bárki meg tudja ezt tenni.

2 .lista

```
#!/usr/bin/perl
use strict;
use DBI;

my $host = “localhost”; # a MySQL elérhetősége
my $database = “vsftpd”; # az adatbázis neve,
➥ahol tároljuk a felhasználókat
my $table = “felhasznalok”; # a tábla neve
➥az adatbázison belül
my $user = “vsftpd_mysql”; # a MySQL felhasználó
➥neve, aki jogosult hozzáférni az adatokhoz
my $passwd = “jelszavunk”; # a MySQL felhasználó
➥jelszava
my $dir = “/usr/share/vsftpd/users”; # a felhasználók
➥egyeni beállításait tartalmazó könyvtár

my $sql = “”; # hagyjuk üresen
my $val = “”; # hagyjuk üresen

if ($ARGV[0] eq “del”) {
    $sql = “delete from $table where nev =
    ➥‘$ARGV[1]’”;
    if ($ARGV[1]) {
        unlink (“$dir/$ARGV[1]”);
    }
} elsif ($ARGV[0] eq “add”) {
    $sql = “insert into $table (nev, jelszo) values
    ➥(‘$ARGV[1]’, password(‘$ARGV[2]’))”;
    open(FH, “>$dir/$ARGV[1]”);
        print “A felhasználó home könyvtára: “;
        $val = <STDIN>;
        print FH “local_root=”.$val.””;
    close(FH);
} else {
    print “Nem megfelelő paraméterezés!\n”;
}

if ($sql) {
    my $dsn =
“DBI:mysql:host=$host;database=${database}”;
    my $dbh = DBI->connect($dsn, $user, $passwd)
        or die “Nem tudtam kapcsolodni
        ➥a szerverhez!\n”;
    my $s = $dbh->prepare($sql);
    $s->execute();
    $s->finish();
    $dbh->disconnect();
}
```

Automatizált felhasználókezelés

Kétségtelen, hogy az első gondolat, ami az emberben felmerül ezzel a megoldással kapcsolatban, az az, hogy milyen jó lenne, ha nem két külön helyen (a *MySQL* adatbázisban, és a */usr/share/vsftpd/users* könyvtárban) lehetne konfigurálni a felhasználókat.
A legegyszerűbb, ha elkészítjük hozzá saját szkriptünket, ami leveszi a vállunkról ezt a terhet is. Íme egy egyszerű *Perl* szkript a feladat megoldására (*2. lista*).
Töltsük ki a megfelelő részeket a beállításainkkal, majd mentsük el például ***user.pl*** néven. Ne felejtsük el beállítani, hogy csak nekünk legyen jogunk hozzáférni ehhez a fájlhoz, mert esetleg kellemetlen meglepetések érhetnek minket. Ha mindezzel megvagyunk, próbáljuk ki, hogy működik-e:

```
debian:~# ./user.pl add kisjozsi jozsi123
```

Válaszoljunk a feltett kérdésre, és ha nem kaptunk hibaüzenetet, akkor nagy valószínűséggel sikerrel jártunk. Próbáljunk bejelentkezni `kissjozsi` felhasználóval.

Törölni a következőképpen tudunk:

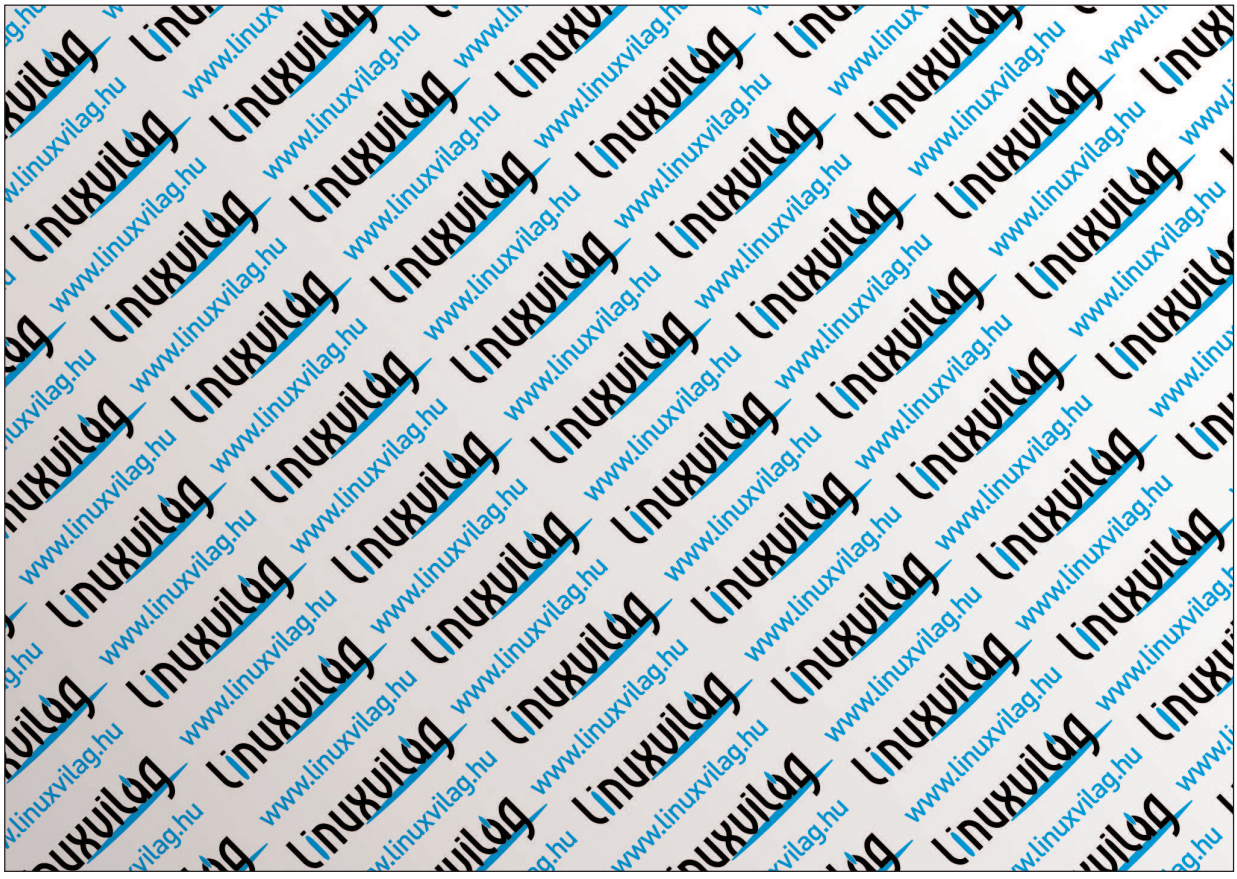
```
debian:~# ./user.pl del kissjozsi
```

A szkript természetesen bővíthető, így más paramétereket is bekérhetünk a felhasználó *home* könyvtárán kívül.

Összegzés

A rengeteg pozitívum mellett szóljunk néhány szót ennek a kivitelezésnek a hátulütőiről is. Kétségtelen, hogy nem a legszebb és legkényelmesebb a *vsftpd* virtuális felhasználó-kezelése. Továbbá abban is biztos vagyok, hogy nem a döbbenetesen precíz sávszélesség-menedzsment funkciója miatt fogják imába foglalni a nevét a közeljövőben. Vagy a józan paraszti ésszel is érthető konfigurációs beállítása miatt…
Arra a feladatra viszont, amire mi most beállítottuk, úgy gondolom, tökéletes. Ha egy kicsivel kényelmesebb lenne a program, az már sajnos a biztonság és a gyorsaság rovására menne. Bár tegyük hozzá, tulajdonképpen egy kis kreativitással a program gyengeségei könnyedén ellensúlyozhatóak (mint azt iménti kis *Perl* szkript is mutatja).
További jó kísérletezést kívánok a *vsftpd*-hez, és amennyiben valakinek kérdése, vagy javaslata lenne az itt leírtakkal kapcsolatban, örömmel veszem, ha megírja azt nekem.

Maróti Tamás (tamas.maroti@async.hu)
18 éves, az ASYNC Magyarország Kft.-nél dolgozik informatikusként, ahol a hálózatbiztonság üzletágat vezeti. Emellett a Leövey Klára KSZKI tanulója. Szabadidejében ha teheti zenél, ír, könyveket olvas, illetve színházi előadásokban statisztál.



Az Ubuntu Linux telepítése

Amint azt egy korábbi számunkban ígértük, ebben a cikkben részletesen ismertetjük a CD mellékletünkön megjelent Ubuntu Linux telepítését és első használatbavételét is.

A *Linuxvilág* olvasói az ***Ubuntu „The Warty Warthog"*** disztribúciójának ***Live*** kiadását kapták kézhez. Ez egy azonnal használatra kész rendszer, hiszen a CD-ről indítva a számítógépet egy teljes grafikus felülettel rendelkező ***Linux*** fogad bennünket. A mostani leírás azonban nem pontosan erre a változatra, hanem a telepíthető korongra épül.

A telepítés

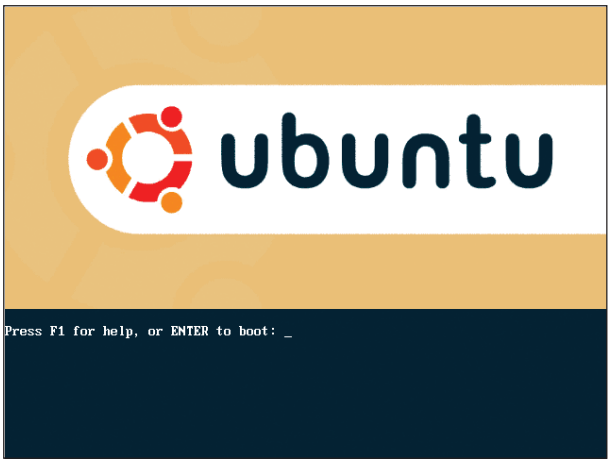
Miután megszereztük a megfelelő ISO lenyomatot az ***Ubuntu*** honlapján található valamelyik tükörszerverről és sikeresen felírtuk egy CD-re, indítsuk el a számítógépet erről a korongról.

Az első kellemes meglepetés: a telepítő magyar nyelven is kiválóan kommunikál. Persze semmi sem lehet tökéletes, amint azt a 2. *ábra* is kiválóan demonstrálja... A szöveg itt arra szólítja fel a kedves felhasználót, hogy távolítson el minden olyan médiát a meghajtókból, amiről a rendszer esetleg elindulhat, majd indítsa újra a gépet. Szerencsére ezek az angolul maradt ablakok vannak kisebbségben.

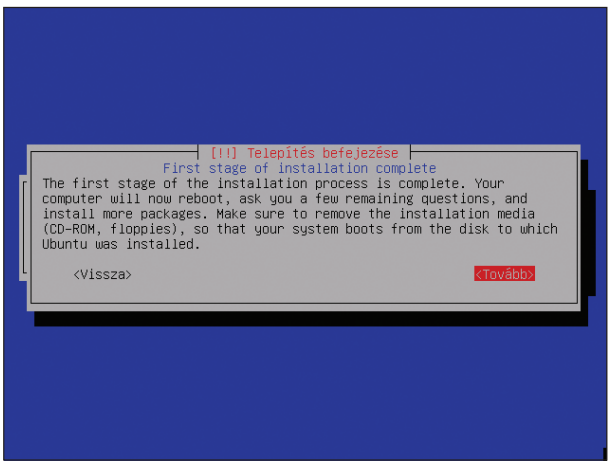
A telepítés első fele mindössze néhány lépés.

Újraindítás után az alaprendszer beállításával kezdjük. Sajnos a képernyő nyelve itt is angol. Az időzóna kiválasztása után, ami a magyar nyelvet választva helyesen ***Europe/Budapest*** értéket kap, létre kell hozunk egy olyan felhasználót, aki elvégezheti az adminisztratív feladatokat. Erre azért van szükség, mert az ***Ubuntu Linuxban*** a root felhasználó alapértelmezésként biztonsági okokból le van tiltva. Így a rendszer karbantartását a `sudo` paranccsal tudjuk végezni, mégpedig a most megadott kitüntetett felhasználó nevében. Szerencsére a fejlesztők odafigyeltek arra is, hogy amikor a ***GNOME*** felhasználókezelőjét indítjuk akkor is a ***sudo*** futtassa a programot. A beállítás folyamán az internetes forrásokat is használatba vehetjük. Ekkor a rendszer automatikusan letölti az ***apt*** forrásokat, majd a frissítést is azonnal elvégzi.

Ez igazán kényelmes, leszámítva persze, hogy jelentősen megnövelheti a telepítés idejét. Nálam 77,4 Mbyte adatot töltött le a rendszer, ami körülbelül 10 percig tartott. A csomagok telepítése közben azok az utolsó beállítások is a helyükre kerülnek, amelyek a grafikus rendszer megfelelő működéséhez szükségesek.



1. ábra Bootolunk...

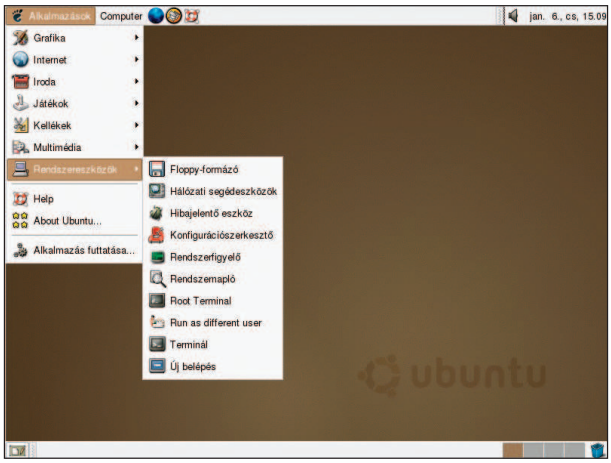


2. ábra Nehéz nyelv a magyar...

Az első használatbavétel

Az Ubuntu a GNOME 2.8-as felületet használja alapértelmezett grafikus környezeteként. Ez szerintem nagyon jó választás volt, mivel ebbe a változatba rengeteg olyan újdonság és hasznos kiegészítés került, amelyek kifejezetten felhasználóbaráttá teszik a rendszert.

3. ábra Fut a rendszer...



4. ábra Rendszerszközök

Nézzük mit is kapunk a *GNOME*-mal:

- Minden fájltípushoz hozzá van rendelve egy alapértelmezett alkalmazás, de választhatunk másik alkalmazást is, abban az esetben, ha egy alkalmazás rosszul regisztrálta volna magát vagy új alkalmazást szeretnénk bizonyos típusokhoz rendelni ,a fájl tulajdonságainál közvetlenül tehetjük ezt meg.
- Az egyik kedvenc szolgáltatásom a DNS-alapú szolgáltatásfelderítés (DNS-Based Service Discovery), amely automatikusan érzékeli a helyi hálózaton található megosztásokat és könnyedén elérhetővé teszi azokat a fájlkezelőn keresztül.
- A rendszereszközök terén is jelentős fejlődés figyelhető meg. Az óra és a hálózat beállításán kívül a felhasználókat és csoportokat is kezelhetjük. A GNOME fejlesztőinek célja, hogy az egyedileg fejlesztett eszközök integrálásra zökkenőmentes legyen. Nyilván ez a záloga annak, hogy a GNOME egy széles körben, sokféle rendszeren használható grafikus felületté válna.

Az igazat megvallva nekem meggyűlt a bajom a `gdm` indításával, ami ezért is nagyon érdekes, mert a `startx` zökkenőmentesen üzemelt a beállításaimmal. És ami még érdekesebb az az, hogy egy egyszerű újraindítással a `gdm` problémája is megoldódott. (Ez a megoldás kicsit hasonlított egy másik rendszerre...)
A 2.6-os rendszermag szerencsére egyre elterjedtebb, tehát nem meglepő, hogy az ***Ubuntu*** fejlesztői is emellett döntöttek. A ***GNOME 2.8***-as egynémely szolgáltatása amúgy eleve csak így érhető el. Maga a felület teljesen egyértelmű, így több szót nem is érdemes rá vesztegetni. Kicsit furcsálltam, hogy az alaprendszernek nem része az `mc`. Úgy tűnik a fejlesztők nem tartják olyan fontosnak ezt a karakteres felületen futó, de igazán remek kis programot, mint én.
Lássuk milyen további eszközök állnak rendelkezésünkre! A ***GNOME Computer-> System Configuration->Device Manager*** menüpontja alatt találunk egy meglehetősen korrekt információkat szolgáltató eszközlistát. Itt szinte minden információ megjelenik ami a */proc* vagy a */sys* fájlrendszerben rendelkezésünkre áll. Szintén ezen az útvonalon érhető el a képernyő felbontását állító program is. Segítségével igazán könnyen „bánhatunk el” a hertzekkel és a pixelekkel. A ***Printing*** menüpontra kattintva érhető el egy nyomtatóbeállító eszköz, aminek a használata meglehetősen magától értetődő. A rendszer által ismert nyomtatókat játszi könnyedséggel vehetjük vele használatba. Ezen kívül beállíthatjuk még a dátumot és időt, itt van a fentebb már említett felhasználókezelés, a hálózati beállítások, ***Synaptic*** csomagkezelő és a ***GDM*** beállító. Végeredményben elmondható, hogy amire egy átlagos felhasználónak szüksége lehet, az itt egy helyen megtalálható.
Maga az ***Unubtu Linux*** akár kezdő felhasználóknak is ajánlható első rendszerként. Tekintettel arra, hogy a fejlesztők deklaráltan nagy hangsúlyt fektetnek a megbízható és minőségi többnyelvűségre, a következő változatokban várhatóan már zökkenőmentes lesz ez a szolgáltatás. Végezetül mindenképpen az ***Ubuntu*** mellett szól a ***Debian*** örökség révén hozzáférhető több ezer azonnal használható csomag.

Kellemes időtöltést!
Csontos Gyula

A programok telepítésének forrásai a /etc/apt/sources.list-ben találhatók

```
deb http://archive.ubuntu.com/ubuntu/ warty main restricted

deb-src http://archive.ubuntu.com/ubuntu/ warty main restricted

deb http://archive.ubuntu.com/ubuntu/ warty universe multiverse

deb http://security.ubuntu.com/ubuntu/ warty-security main restricted

deb-src http://security.ubuntu.com/ubuntu/ warty-security main restricted

deb http://ubuntu-bp.sourceforge.net/ubuntu/ warty-backports main universe
```

MOODLE: Egy ingyenes e-learning keretrendszer (2. rész)

Az előző részben sikeresen telepítettük a Moodle rendszert saját gépünkre, megismerkedtünk annak alapvető funkcióival. Most további lehetőségeket kívánok ismertetni a rendszerről, építve az előző rész tartalmára.

Az előző részben inkább a ***Moodle*** rendszer számára szükséges környezet megteremtése volt hangsúlyos, és magáról a rendszerről kevesebb szó esett. Azért most inkább magára a keretrendszerre, és annak további lehetőségeire szeretnék koncentrálni, bár az előző részben megemlítettem, hogy hogyan tudunk tananyagokat, és tesztkérdések felvinni, érezzük, hogy az elektronikus oktatás ennél többről szól.

A hagyományos és az elektronikus oktatás közötti párhuzam

Az oktatás az iskolában folyik, a tanárok, tanulók bemennek az iskolába, és tanórák keretében a tanárok oktatják a tanulókat. Az elektronikus oktatás is hasonló, az iskola jelen esetben a honlap, ahová a tanárok, tanulók virtuálisan ellátogatnak. Az iskolának van egy igazgatója, mely intézi az iskola ügyeit, esetleg ő is oktathatja a tanulókat, tanárokat. Jelen esetben az igazgató a honlap adminisztrátora, aki karbantartja a honlapot, új tanárokat, tanulókat vehet fel, ha a tanároknak gondjuk van valamivel, akkor hozzá nyugodtan fordulhatnak. Az oktatott tananyag sokféle lehet, az egyes tantárgyakat pedig még kisebb részekre tudjuk osztani, például a történelem esetében (őskor, ókor, középkor, stb.).
A ***Moodle*** rendszerben a legnagyobb egység a tanfolyam kategória, mely tanfolyamokat tartalmaz, amelyeket a tematikájuk szerint még további részekre van lehetőségünk felbontani. Egy tananyagot csak olyan tanárok taníthatnak, melyeknek megvan a megfelelő képesítésük, ez az elektronikus oktatásban sincs másképpen, a jogosultságok kezelésével lehetőségünk van megoldani, hogy csak bizonyos tanárok módosíthassanak egy adott tananyagot. Az oktatásnak csak akkor van értelme, ha a tanulók és a tanárok is meg tudnak bizonyosodni arról, hogy mennyire sikerült elsajátítani a tananyagot, ezt természetesen elektronikus formában is meg lehet tenni különféle teszt kérdésekre adott válaszok segítségével.
Természetesen a tanárok színesebbé is varázsolhatják az órát képekkel, hangfelvételekkel, vagy videó bejátszásokkal, és ez az elektronikus oktatásban sincsen másképpen. Ha azt hisszük, hogy az eddig felsorolt rendszer már egy iskola, akkor tévedtünk, ugyanis hiányzik még egy nagyon fontos elem, a tanárok, és a tanulók közötti párbeszéd. Szükséges, hogy a tanulók vissza tudjanak kérdezni egy tananyagrészre, amennyiben az nem elég világos számukra, szükséges, hogy a tanulók egymással is tudjanak beszélgetni.
A ***Moodle*** megvalósítja ezeket a kívánalmakat, talán még egy kicsit többet is ennél, de sajnos van egy nagy hátránya, ez pedig az hogy az emberek soha sem lépnek személyesen egymással kapcsolatba. Pontosan ezért az elektronikus oktatás nem helyettesítheti a hagyományos általános és középiskolai iskolarendszert, hanem csak azzal együtt, azt kiegészítve, bővítve annak lehetőségeit használható.

Az elektronikus oktatás jövője

Az elektronikus oktatást a jövőben nem csak a hagyományos középiskolai, felsőoktatási intézmények használhatják, hanem ennél sokkal szélesebb felhasználói kör számára nyújt majd segítséget. Gondoljunk csak bele abba a helyzetbe, amikor egy nagy vállalat, mely több telephellyel rendelkezik az ország területén új szoftvert vezet be. Az új szoftver használatát pedig valamilyen formában oktatni kell. Nyilván nehezen oldható meg, és sok plusz költséggel jár, hogy minden egyes telephelyen külön-külön oktassuk a kollégákat az új szoftver használatára. A központosított oktatás pedig a sok utazás miatt elég nehézkes, a személyes részvétel sok felesleges utazást kíván meg a kollégáktól. Az elektronikus oktatásnak nagyon sok előnye van a hagyományos oktatással szemben, hiszen a kollégák saját szabadidejük szerint képesek végigmenni az egyes tananyagokon, ha valaki elmegy kisebb szabadságra, akkor sem marad le egyetlen tananyagról sem, hiszen később könnyedén pótolni tudja, illetve később az újonnan belépő kollégák is könnyedén megismerhetik a szoftvert az elektronikus tananyagokon keresztül.

Tanári jogosultságok kezelése

Az előző részben sikeresen telepítettük a ***Moodle***-t, létre tudtunk hozni tanfolyamokat, és tananyagokat, illetve tesztkérdéseket is tudtunk készíteni az egyes tananyagok végén. Viszont most lépjünk tovább, és engedjük meg mások számára is, hogy tananyagokat helyezhessenek el az oldalunkon, megnyitva ezzel a kaput a tanári kar számára. A főoldalról elin-

1. kódrészlet

```
var $CharSet = “iso-8859-2”; //magyar ékezetes karakterek
var $From = “admin@enelsomoodleoldalam.hu”; //ez lesz a levél feladójának a címe
var $FromName = “Enelsomoodleoldalam Admin”; //ez lesz a feladó neve
var $Sender = “admin@enelesomoodleoldalam.hu”;
var $Mailer = “smtp”; //smtp szerveren keresztül küldjük majd az e-mailt
//Az smtp szerverre vonatkozó beállítások
var $Host = “smtp.sajatszerverem.hu”;
```

dulva az ***Adminisztráció/Felhasználók/Új felhasználó*** hozzáadása útvonalat végigjárva tudunk új felhasználókat hozzáadni a *Moodle* rendszerhez, később látni fogjuk, hogy megoldható az is, hogy a tanulók maguk jelentkezzenek be a *Moodle* rendszerbe az e-mail címük segítségével. Miután minden adatot kitöltöttünk az új felhasználóról, akkor a ***Profil frissítése*** gombra kattintással rögzíthetjük az adatbázisban. Ilyen módon később is képesek vagyunk felhasználókat adni a rendszerünkhöz. A hozzáadott felhasználók mindig csak diák státuszt kapnak, ezért ha azt szeretnénk, hogy ezek a felhasználók taníthassanak is, akkor azt máshol kell beállítanunk. Adjunk hozzá a rendszerünk tehát pár felhasználót, lehetőleg ők legyenek a tanárok, majd pedig készítsünk egy új tanfolyamot a már megismert módon.
Az előző cikkben erre nem tértem ki, de miután kitöltöttük, és elmentettük a tanfolyam adatait, azután a *Moodle* automatikusan megkérdezte tőlünk, hogy milyen tanárokat szeretnénk hozzáadni a tanfolyamhoz. Mivel akkor még nem voltak tanár jelöltjeink, ezért ezt a lépést átugrottuk, most viszont lehetőségünk van megadni a tanfolyamhoz tartozó tanárokat, miután kiválasztottuk őket kattintsunk a ***Változás mentése*** gombra, hogy a beállítások megmaradjanak. A jogosultságokat természetesen később is módunkban áll megváltoztatni, melyet az adott tanfolyam megnyitása után az ***Adminisztráció*** doboz ***Tanárok*** menüpontjában tehetünk meg.

Tanulók regisztrációja e-mail cím alapján

Kicsit fáradságos lenne minden egyes tanulót mindig magunknak felvenni a rendszerbe, ezért meg kell engednünk, hogy a felhasználók regisztrálhassák magukat. Célszerű viszont ilyenkor egy e-mail címet is kérni mindegyik regisztrált felhasználótól, így elkerülhető, hogy egy személy többször regisztrálja magát, illetve a megadott címen kapcsolatban is tudunk lépni a tanulóval. Az e-mail alapú regisztráció úgy történik, hogy a tanuló megadja a saját e-mail címét a regisztráció alkalmával, majd a megadott címre a szerver egy egyedi linket küld, a linkre való ellátogatással a szerver regisztrálja az adott felhasználót.
Regisztráció után lehetőség nyílik a vendég felhasználóknak nem látható oldalak megtekintésére is, tesztek kitöltésére és későbbi visszakeresésére is van mód. A *Moodle* a *PHPMailer* segítségével küldi az üzeneteket a kívánt címzettnek, ezért állítsuk be a *PHPMailer*-t, hogy a megfelelő csatornán keresztül küldje el az e-mailt.
A *PHPMailer* segítségével használhatjuk a *mail*, *sendmail*, vagy az *smtp* szolgáltatásokat, az egyszerűség kedvéért mi használjunk egy *smtp* kiszolgálót, melynek nem feltétlenül a *Moodle*-el azonos szerveren kell lennie, használjuk például az internetszolgáltatónk, vagy a helyi hálózat szerverét, legyen ennek címe ***smtp.sajatszerverem.hu*** A beállítások a *Moodle* könyvtárán belül a ***lib/phpmailer/class.phpmailer.php*** fájlban találhatóak. Egy példa beállítást találhatunk az *1. kódrészletben*, a többi beállítást célszerű meghagyni, és valószínűleg jó is lesz nekünk. A beállítások után célszerű kipróbálni, hogy jól működik-e ez az opció.

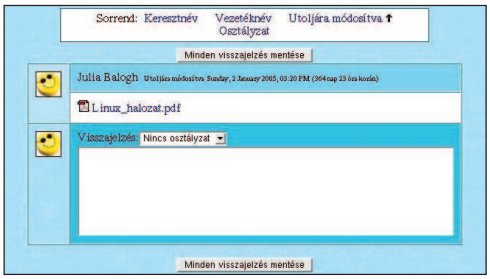
Komplex feladatok kezelése

Az cikk előző részében megemlítettem, hogy a *Moodle*-lel képesek vagyunk ellenőrizni, hogy a tanulók mennyire sajátították el a tananyagokat, ellenben lássuk be, hogy eldöntendő, vagy többszörös választáson alapú kérdésekkel nagyon sok mindent nem lehet ellenőrizni, például ilyen a verselemzés, amikor egy esszét várunk el a tanulóktól, vagy például, ha a Gimp programot oktatjuk, akkor célszerű lenne a tanulók tudását úgy lemérni, hogy egy képet várunk tőlük.
A *Moodle* szerencsére támogatást biztosít nekünk ehhez, vagyis lehetőségünk van kiírni egy feladatot a tanulók számára, és a tanulóknak egy fájlt kell feltölteniük, mely a megoldást tartalmazza. Természetesen ez egyelőre semmivel sem nyújt többet annál, hogy a tanuló elküldi a saját megoldását nekünk e-mail csatolásként. Vagy mégis? Először is lehetőségünk van határidőt megadni, tehát nem kell magunknak azzal foglalkozni, hogy valaki a határidő után küldte be a megoldását, illetve lehetőségünk van letiltani azt, hogy valaki a már beküldött megoldását újból beküldje.
A beküldött megoldásokat természetesen a *Moodle* keretein belül képesek vagyunk értékelni, sőt megjegyzést is írhatunk az értékelésünkhöz, melyet csak az adott hallgató képes megtekinteni. Könnyűszerrel láthatjuk azokat a tanulókat is, akik még nem küldték be saját megoldásukat.
Most pedig nézzük hogyan is tudunk létrehozni egy ilyen feladatot. Hasonlóan a *Quiz*-hez válasszuk ki a ***Feladat***, vagy ***Journal*** menüpontot az ***„Add an activity...”*** legördülő menüből, majd pedig töltsük ki megfelelően a szükséges paramétereket, és mentsük el a változtatásunkat. Ha pedig valaki beküldött egy megoldást, akkor tanári jogosultságokkal megnyitva a feladatot, a feltöltött fájlokat és beküldőiket a *1. ábrához* hasonlóan tudjuk megjeleníteni, és ugyanezen a lapon tudjuk értékelni is a megoldásokat.

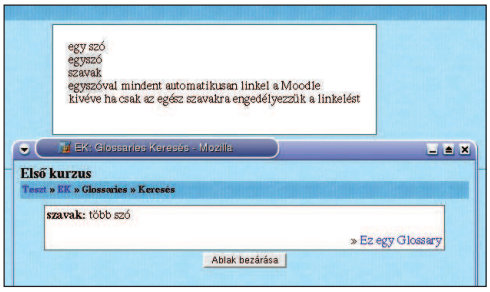
Fórumok és csevegés

A fórumok igen közkedvelt részei a weboldalaknak, hiszen lehetőséget biztosítanak arra, hogy távol lévő felek megvitassanak valamit, véleményt nyilvánítsanak valamiről, vagy csak egyszerűen arra, hogy beszélgessenek egymással.

A *Moodle* keretrendszerben mi is egyszerűen tudunk fórumokat készíteni, ehhez be kell kapcsolnunk a szerkesztést, és a megfelelő helyen a tanfolyamon belül a legördülő menüből válasszuk ki a ***Fórum*** hozzáadását. Ezután töltsük ki a fórum paramétereit, majd mentsük el a változtatásokat. Egy fórumon belül lehetőségünk van több témát is indítani, így kategorizálhatjuk a hozzászólásokat. A fórumok részletesebb beállításaira nem térnék ki. Viszont a fórumok nem nyújtják a felhasználóknak a tényleges párbeszéd érzetét, ezért a *Moodle* lehetőséget biztosít a tanulók közötti on-line kommunikációra is a ***chat modul (csevegés)*** segítségével. Új *chat*-et az egyes tanfolyamokhoz tudunk adni, ha bekapcsoljuk a tananyag frissítését, és a legördülő menüből kiválasztjuk a *chat* hozzáadása opciót. A *Moodle*-be integrált *chat PHP* alapú, ezért ne várjuk el tőle a ma oly divatos Java alapú *chat* kliensek sebességét, de 20-30 fő egy szobát mindenképpen élvezetesen tud használni, viszont mindenképpen pozitívum, hogy *chat* esetében is a hozzászóló által feltöltött kis fénykép látszik, ezáltal egy picit emberközelibb a „szövegelés”.



1. ábra A beérkezett megoldások megtekintése, és értékelése



2. ábra Példa a magyarázó szövegek automatikus linkelésére a tananyagokban

A HTML szerkesztő testreszabása

Amennyiben a *Moodle* beépített *HTML*-szerkesztőjét használjuk a tananyagok szerkesztéséhez, akkor elengedhetetlen lesz számunkra, hogy a beépített szerkesztő néhány alapvető tulajdonságát módosítsuk. Az ***Adminisztráció*** doboz ***Beállítások/Editor settings*** menüpontjában állíthatjuk át a beépített böngész legalapvetőbb tulajdonságait.
Az ***editorfontfamily*** menüpontban tudjuk meghatározni azt, hogy a beépített *HTML*-szerkesztő milyen fontokkal dolgozzon, illetve, hogy a legördülő menüben a fontok milyen sorrendben kövessék egymást. Célszerű az általunk használni kívánt fontot a lista elejére tenni, mert ekkor minden egyes új tananyag felvitelénél rögtön ezzel a betűtípussal tudjuk kezdeni felvitelt. Az oldal alján lehetőségünk van megadni az egyes fontok preferenciát is, vagyis ha mi például *Verdana* fontot használunk az oldalunkon, akkor annak a kliensnek, akinek nincs telepítve a *Verdana* fontkészlete, a böngészője milyen alternatív fonttal próbálja meg megjeleníteni az oldalt. *Verdana* font esetén a sorrend például így néz ki: ***verdana,arial,helvetica,sans-serif***, de lehetőségünk van saját sorrendet is definiálni.

Szójegyzék, avagy szó- és fogalommagyarázat

A felvitt tananyagban néha elkerülhetetlen, hogy szakszavakat, szakkifejezéseket használjunk, de ezeket megmagyarázni minden egyes előfordulásukkor felesleges volna, az sem biztos, hogy jó megoldás, ha az első előfordulásukkor magyarázzuk csak el jelentésüket, hiszen lehetnek olyan tanulók, akik csak bizonyos anyagrészekre kíváncsiak, és így nem fogják megtalálni a szó pontos magyarázatát. A *Moodle* keretrendszer lehetőséget biztosít arra, hogy a szavakat szótárba gyűjtsük és jelentésüket ott elmagyarázzuk, ez a szótár bárhonnan elérhető és az adott szó magyarázatát csak egyszer kell felvinni.
Ez nyilván kellemes szolgáltatás, de a *Moodle* ennél többre képes. Lehetőséget biztosít arra, hogy ha a tananyagban egy a szótárban szereplő szó található, akkor az adott szó automatikusan egy hivatkozás legyen a szó magyarázatára a szótárban. Állítsuk be tehát ezt a szolgáltatást.
A Főmenü legördülőmenüjéből válasszuk ki a ***Glossary*** hozzáadása opciót, majd pedig válasszuk ki a ***Glossary*** típusánál a ***„Main Glossary”*** opciót, ezáltal egy globális az egész *Moodle* honlapra érvényes szójegyzéket készítünk, a név és rövid leírás megadása után mentsük el a változásokat, és kapcsoljuk be a *Moodle* rendszerben az automatikus linkelést a szójegyzék szavainak használatakor. Ezt az ***Adminisztráció/Beállítások/Filters*** menüpont alatt tehetjük meg, ha eddig nem lett volna bekapcsolva a ***Glossary Auto-linking***. Az előbb hozzáadott szójegyzékre kattintva a főoldalon láthatjuk, hogy még üres, ezért adjunk hozzá néhány bejegyzést, hogy megtapasztaljuk a tényleges működést. A szavak felvitele közben lehetőségünk van arra, hogy megtiltsuk, illetve engedélyezzük az aktuális szónak az automatikus linkelését, beállíthatjuk, hogy a linkelés figyelembe vegye a kis- és nagybetűk közötti különbséget, illetve, hogy a szó teljes egyezését követelje meg.
A fenti példában (2. *ábra*) a szótárba az „egyszó” (nyelvtanilag hibásan, egybeírva) került bele, úgy, hogy a rendszer automatikusan linkelje, illetve ne csak az egész szavak egyezését figyelje, ennek megfelelően az „egyszóval”, „egyszó” is természetesen link lesz, de az „egy szó” nem. A „szavak” úgy került bele a szótárba, hogy az automatikusan linkelődjön, de csak akkor ha az egész szó egyezik, ezért például a „szavakra” szó nem link, de a „szavak” természetesen link. A linkre kattintva a magyarázat egy felugró ablakban jelenik meg, ahogyan az a 2. ábrán is látszik.

Utószó

Azok számára, akik komolyabban érdeklődnek a rendszer iránt bátran ajánlom az angolul megírt tanári segédletet (➲ http://moodle.org/docs/teacher-manual.pdf.zip), mely körülbelül 80 oldalon keresztül részletes illusztrációkkal mutatja be a keretrendszer használatát. Ajánlom még a keretrendszer honlapján található (➲ http://www.moodle.org), a regisztrált felhasználók számára ingyenes fórum lehetőségét is, ahol sokféle kérdésre választ kaphatunk.

Horváth Ernő

A felhasználói viselkedés vizsgálata (5. rész)

Cikksorozatom utolsó részében a bemutatott lehetőségek fényében néhány mérést készítek el a saját fejlesztésű programom segítségével. Végül a mérési eredményekből levonható következtetések és a gyakorlati jelentőség elemzésével lezárom a cikksorozatomat.

Eseménynapló források szerkezete, előzetes ismeretek

A mérések elvégzéséhez szükséges eseménynapló fájlok beszerzésénél megpróbáltam a következő szempontokat figyelembe venni:

- Legyen egy olyan eseménynapló, amely tartalmaz munkamenet azonosítókat is. Így lehetőség nyílik az összehasonlításra is: mennyit tévedünk amikor idő alapon különítjük el a látogatóinkat.
- Legyen olyan szerkezetű eseménynapló, amely keletkezéséről (a naplózott portálról) a lehető legtöbb információval rendelkezem. Így bizonyos kiugró adatokra könnyebben lehet magyarázatot találni.
- Szükséges egy nagyméretű, ismeretlen szerverről származó eseménynapló az általános, független következtetések felállításához.

A felhasznált három különböző eseménynapló eltérő portáloktól származik. Mind látogatottságban, mind tartalomban, mind a portálok szerkezetében jelentősek az eltérések. A legkisebb méretű, mindössze 15 MB-os eseménynapló egy apróhirdetéseket kezelő weboldal szerveréről származik. A látogatók legjellemzőbb tevékenysége a keresés az apróhirdetési adatbázisban. A portál összesen 19 jól elkülönülő weboldalból áll (keresés, eredmény lista, felvitel, impresszum, stb.), minden egyes weblap funkciója jól meghatározható. Ez az egyetlen eseménynapló, amely munkamenet azonosító adatokkal rendelkezik. Az eseménynapló összesen 77.562 soros, időben 5 és fél hónap látogatásait tartalmazza. A második szempontnak leginkább megfelelő portál eseménynaplója már lényegesen összetettebb, tekintve, hogy egy eseménynaplóban több különböző portál verzió bejegyzései is benne vannak. Ez egy online folyóirat, rengeteg (több mint 600) letölthető dokumentummal. A legjellemzőbb használat a dokumentumokban történő online keresés (adatbázis segítségével). Ez a portál is jól elkülönülő fájlokból épül fel, de a több, jelentősen eltérő verzió megjelenése egy naplófájlban megnehezíti az analízist. Megfelelő szűrési paraméterekkel a régebbi verzió bejegyzései könnyen kihagyhatóak. Az eseménynapló 93 MB, összesen 430 ezer bejegyzést tartalmaz alig 2 hónapos időtartam alatt.
A legnagyobb és legösszetettebb eseménynapló egy ismeretlen webszerverről származik. Bármiféle vizsgálat előtt az eseménynapló „kézi" átvizsgálásával a következők állapíthatók meg:

- tartalmazza egy galéria lekéréseit rengeteg képpel,
- tartalmazza egy olyan lekéréseit, amelyre jellemző a GET paraméterbe kódolt weboldal megkülönböztetés,
- továbbá tartalmaz sok egyéb letölthető dokumentációt (több szoftver online dokumentációjának tükrözése), amelyre jellemző a sok különálló fájl.

Az eseménynapló egy heti forgalmat tartalmaz, 721 ezer sora 149 MB méretű.
Az analíziseket egységesen egy ***Intel Pentium4 2,60 GHz*** processzorral, 512 MB memóriával felszerelt számítógépen hajtottam végre. Az analízis folyamatok futtatása során törekedtem az egységes szoftverkörnyezet biztosítására.

Különböző paraméterekkel végzett vizsgálatok

Első lépésben minden esetben az összes különböző lekérést kell az eseménynaplókból kiolvasni. Ez azért szükséges, hogy előzetes ismeretek nélkül is tudjuk alapszinten paraméterezni az előszűrési feladatokat. Mivel az eseménynaplókról és a hozzájuk tartozó portálokról nem minden esetben rendelkeztem elégséges információkkal, ezért minden modellkészítés során az 1% alatti elemeket elhagytam. Ez azt jelenti, hogy a ritkán előforduló sorokat nem vettem figyelembe.
Az apróhirdetési portálról származó naplófájl az egyetlen, amely munkamenet azonosítókat is tartalmaz. Először a különböző kéréseket kértem le, amely eredményből jól látszik, hogy ez a portál egyszerű szerkezetű, kevés weboldalból áll, és az egyes aloldalak funkciója is jól elkülönül. Jól látható az is, hogy a portál összes lekérésének (kb. 23 ezer) közel 89%-a 4 aloldal között oszlik el.
A munkamenet azonosítók alapján elvégzett analízis szerint összesen 9672 látogató járt a portálon, vagyis egy átlag látogató 2.1 weboldalt töltött le, ami összhangban van azzal, hogy az összes lekérés 75%-a két aloldalra irányult.
Az *1. táblázatban* látható adatok jelentése:

- Felhasználói útvonalak: a különböző időbeállításokkal vagy munkamenet azonosító felhasználásával lefuttatott analízis során elkülönítésre került látogatók száma.

1. táblázat *Apróhirdetési portál statisztika*

	munkamenet	10 perc	30 perc	60 perc	Átlag (382 sec)	Átlag 2x (764 sec)
Felhasználói útvonalak	9672	9261	8981	8857	9393	9189
Különböző útvonalak	23	18	16	14	18	18
Azonos oldalak egy látogatáson belül	17	12	10	8	12	12
Belépő oldalak	15	15	15	15	15	15
Kilépő oldalak	15	14	14	14	14	14
Átkattintási idők száma	3	3	3	3	3	3
Átkattintási idők átlag (max) sec	382	75	172	292	51	94
Átkattintási idők maximum (max) sec	40743	601	1756	3595	382	760

2. táblázat *Folyóirat portál statisztika*

	10 perc (1%)	10 perc (0,5%)	30 perc	60 perc	Átlag 2x (410 sec)
Felhasználói útvonalak	6785	6785	5857	5515	7265
Különböző útvonalak	4	13	10	9	11
Azonos oldalak egy látogatáson belül	3	9	6	6	7
Belépő oldalak	15	21	21	24	15
Kilépő oldalak	9	16	16	17	14
Átkattintási idők száma	9	15	16	17	3
Átkattintási idők átlag (max)	159	159	191	205	51
Átkattintási idők maximum (max)	600	600	1794	3600	382

- Különböző útvonalak: az összes látogató útvonalában ennyi olyan különböző útvonal van, amelyek gyakorisága a küszöbérték (1%) felett van.
- Azonos oldalak egy látogatáson belül: a különböző útvonalak között előforduló olyan weboldal csoportok száma, amelyek az útvonalon belül ugyanazokat a weboldalakat tartalmazzák, itt a sorrendjük nem számít.
- Belépő és kilépő oldalak: azt mutatja meg, hogy hány olyan weboldal van, amelyre illetve amelyről a küszöbérték feletti számban léptek tovább / érkeztek a látogatók.
- Átkattintási idők száma: azoknak a weboldalaknak a száma, amelyekről a küszöbérték feletti számban történt átkattintás egy másik, portálon belüli oldalra.
- Átkattintási idők átlaga: az összes átkattintási lépésnél átlagolásra kerül a két lekérés között eltelt idő, ez az érték másodpercben mutatja ezen átlagidők maximumát.
- Átkattintási idők maximuma: a legnagyobb olyan idő, amely két lekérés között eltelt.

A fenti táblázatban látható, hogy a különféle idő paraméterek beállítása ellenére sem térnek el több mint 10%-kal a felhasználói útvonal adatok. Ez azt jelenti, hogy a látogatók 90%-a kevesebb, mint 10 percenként (legkisebb időparaméter) lép egyet a portálon belül.
Jól látható, hogy a be- és kilépő oldalak és az átkattintási idők számai nem változnak. Ennek ez azért van, mert a látogatások nagyobbrészt egy jól meghatározott oldalcsoporton belülre irányulnak. Ezek a változatlanságok azt is jelentik, hogy az összes lekérés között ennyi jellemzően használt van. Kirívó adat a munkamenet azonosítóval történő felhasználó elkülönítésnél kapott, az átkattintási idők maximum értékének legnagyobb értéke: 40743 másodperc, közel 11 és fél óra. Ez egy kirívó eset, keletkezése úgy lehetséges, hogy a felhasználó böngészője a reggeli megnyitáskor kapott munkamenet azonosítót az esti visszatérésig nem hatástalanította, és azt újra elküldte a szervernek. Véleményem szerint ez egy böngésző hiba.
Egyenként összehasonlítva a különböző útvonalak és az azokon belüli azonos oldalak analízis eredményeit megállapítható, hogy az eredmény méretében az eltérés a felhasználói útvonalak száma miatt változik, a küszöbérték szerinti elhagyás előbb vagy később dobja el az eredmények egyes részeit.
Az on-line folyóirat portál eseménynaplójának analízise nehezebbnek bizonyult. A bevezetőben említett 1%-os küszöbérték esetén az egyes modellek csak néhány adatot tartottak meg, ezért itt lejjebb kellett venni a vágási értéket 0,5%-ra, vagyis a portál egy jól meghatározható részén van csak érdemi látogatottság. Az analízis eredmények statisztikája olvasható a 2. *táblázatban*.
Az apróhirdetési portálhoz képest itt az elkülönített látogatók száma szélesebb skálán mozog (15% illetve 23%), ami azt jelzi, hogy a látogatók nagyobb hányada maradt hosszabb ideig a portálon.

A különböző útvonalak és az azokon belüli azonos oldalak analízis eredményeit megvizsgálva elmondható, hogy a látogatók jellemzően néhány weboldal köré csoportosulnak: főoldal, keresés, tartalomjegyzék, cikk olvasás. Ez a megállapítás egybevág az összes különböző lekérésben található adatokkal, mert az összes lekérés 75%-a erre a néhány oldalra irányul.

Az ismeretlen webszerver eseménynaplójának vizsgálata az említett több portálos szerkezete miatt nehézkes. Ugyanakkor a fentiekhez hasonló eredményeken kívül megmutatkozott az analízis egy újfajta felhasználási lehetősége: a be- és kilépő oldalak statisztikájának vizsgálatával egyértelműen megállapítható volt, hogy a naplófájl több jól elkülönülő weboldal (portál) lekéréseit tartalmazza, azaz a be- és kilépő oldalak partíciókra estek szét. Jellemzően az egyes partíciókba tartozó oldalak szinte kizárólag az azonos partícióban lévőkre hivatkoztak. Az egyes partíciókból a portálok szerkezetének ismeretében könnyen felállíthatóak azok az előszűrési feltételek, amelyek mentén szétbontva az eseménynaplót, a keletkező résznaplók, mint az egyes portálok önálló eseménynaplói könnyebben és pontosabban analizálhatóak.

Összehasonlítás

A fenti eredményeket összehasonlítva más, hasonló képességű programmal (***Advanced Log Analyzer***, ***http://www.abacre.com/ala/***) az eredményekről elmondható, hogy a szoftverek által előállított modellek hasonlítanak egymásra. Például mindkettőben ugyanazok a weboldalak jelennek meg az eredményekben az első helyeken. Ugyanakkor egyes számszerű értékekben eltérések vannak.

A lekérések számában az eltérés mértéke minimális, a különbség az összevonások és egyszerűsítések, illetve a könyvtár lekérések kezeléséből adódik. A be- és kilépő oldalak statisztikáinál az eltérés valószínűsíthető oka a könyvtár lekérések kezelése, ugyanis az ***Advanced Log Analyzer*** teljesen külön kezeli a könyvtár és weboldal lekéréseket, míg a saját fejlesztésű szoftver képes a megfelelő bejegyzéseket jelentésileg azonosnak tekinteni (tipikus példa: ***http://www.pelda.com/*** azonos a ***http://www.pelda.com/index.php*** lekéréssel).

A felhasználók számában és útvonalaikban az eltérés magyarázata nem triviális, ugyanis az ***Advanced Log Analyzer*** eredményei önmagukban is ellentmondóak: míg a felhasználói profiloknál 5238 látogatót említ, addig a felhasználók által eltöltött idő statisztikáknál már csak 3071 látogatásról van szó (az adatok az első, apróhirdetésekkel foglalkozó weboldalra vonatkoznak).

A munkamenet azonosítóval készült eredményeket etalonnak tekintve a felhasználók számában az ***Advanced Log Analyzer*** eltérése közel 50%-os. Véleményem szerint a különbség egy jelentős részére magyarázat lehet a könyvtár és oldal lekérések összevonásának hiánya. További eltérést okozhat a paraméterezhetőség különbözősége és az ***Advanced Log Analyzer*** idő paraméterének ismeretlensége.

Gyakorlati jelentőség, felhasználhatóság

Az eredmények ismeretében mindössze az eseménynaplók felhasználásával képet lehet alkotni az adott portált felkereső látogatók mozgásáról, így akár régóta üzemelő weboldalak is bevonhatóak az analízisbe.

A felhasználói útvonalak önmagukban már csak méretüknél fogva sem alkalmasak következtetésekre. Ezt az eredményt minden esetben még fel kell dolgozni, hogy az általánosságok is a felszínre kerüljenek.

A különböző útvonalak statisztika elárulja, hogy melyek a legjellemzőbb bejárási útvonalak az adott portálon.

Az adatok segítségével ezeket a gyakori látogatású útvonalakat megfelelően lehet optimalizálni mind marketing szempontokból (például drágábban árult reklámfelület, célzott reklám), mind a szerver terhelésének optimális elosztására (például a frekventált weblapok gyorsítótárazása).

Az adatok ismeretében lehet átalakítani a portál belső szerkezetét is (a látogató előbb jusson hozzá a kívánt információhoz). Hasonló eredményeket és lehetőségeket ad az azonos oldalak egy látogatáson belüli eloszlása is.

Az átkattintási idők vizsgálatával pontosabb kép nyerhető nemcsak a látogatók útvonalairól, hanem az egyes weboldalakon eltöltött idők megoszlásáról is. Egy lehetséges felhasználás: azokat a weboldalakat, ahonnan jellemzően nagyon rövid idő alatt elkattintottak a látogatók, valamilyen módon át kell alakítani, vagy a tartalom módosításával, vagy akár magának a portál szerkezetének módosításával, hiszen az oldal a látogatók számára nem érdekes.

A belépő oldalak statisztika leginkább arra jó, hogy a látogatók portálba lépési pontjait megtaláljuk. A leggyakoribb belépési pontokon megfelelő szolgáltatást kínálva a látogatót sokkal nagyobb valószínűséggel lehet a portálon megtartani, így növelve a látogatottságot és az azzal járó előnyöket (például reklámbevételek, eladások növelése).

A kilépő oldalak statisztika segítségével találhatóak meg azok a weboldalak, amelyek után a felhasználó elnavigál a portálról. Ezen weboldalak tartalmának ismeretében a portál működtetői eldönthetik, hogy normális távozásról van-e szó (például egy címgyűjtemény esetén), vagy a nem megfelelő tartalom miatt hagyják el gyakran az adott oldalon keresztül a portált a látogatók.

A különböző kérések eredménye egyértelműen megmutatja, hogy melyek a portál leggyakrabban látogatott oldalai. Általánosságban elmondható, hogy a különböző analízisek során előálló adatok hatékony felhasználásához szükséges a vizsgált portál beható ismerete úgy tartalmilag, mint szerkezetileg. Fontos továbbá a látogatókról más forrásból is információkat szerezni (például kérdőíves megkérdezés), hogy a többféle eredmény összevetésével az optimális módosításokat lehessen megvalósítani.

Úgy vélem, hogy egy portál hosszú távú és eredményes üzemeltetéséhez elengedhetetlenül szükséges a folyamatos karbantartás és fejlesztés. Ezen folyamat egyik segítője lehet az általam bemutatott vizsgálati módszer: segítségével naprakészen tudjuk követni a látogatóink szokásainak változását.

Beszédes Balázs (beszedes@ei.hu)
24 éves, az e-Média Informatikánál mérnök-informatikus. Hobbija a kerékpározás és a kirándulás.

Samba – Windowsban is otthon (2. rész)
Tartományvezérlők beállítása

A cikksorozat előző részében megpróbáltam ízelítőt adni abból, minként lehet egy nyílt forráskódú, ingyenes rendszert egy drága üzleti szoftver kiváltására használni. A sorozat első részét tekinthetjük egy ízelítőnek, kedvcsinálónak ahhoz, hogy komolyabban foglalkozzunk a Sambával.

Aki végigolvasta az előző cikket, az a végére érve olyan tudás birtokában volt, amellyel az első lépéseket megtehette, készíthetett egy nagyon egyszerű állománykiszolgálót, valamint kellő alapot – és remélem motivációt – kapott, amelyre a továbbiakban már bátran építkezhetünk.
Ahhoz, hogy egy olyan rendszert építsünk, amelyik megállja a helyét üzleti környezetben is, nem elég pusztán egy olyan konfigurációt felállítani, ahol vannak kiosztások, és ahhoz bizonyos felhasználók hozzáférhetnek. Vállalati környezetben ennél többre van szükség. Szeretnénk a hálózat minden kiszolgálójához azonos felhasználói paraméterekkel hozzáférni és szeretnénk ezen paraméterek beállításait és tárolását központosítani. Szeretnénk, ha a felhasználók a belépéskor a rendszerben olyan és csak olyan jogokat szerezhetnének, amelyekre szükségük van, illetve szeretnénk azt is biztosítani, hogy a felhasználók a hálózat bármely munkaállomását is vegyék igénybe, azonos munkakörnyezetet kapjanak. Erre a *Windows* világában a ***tartománykezelés (domain control)*** a megoldás. Mivel jelenleg az a célunk, hogy a *Windows* munkaállomásainkat összeházasítsuk a linuxos kiszolgálóinkkal, ezért nekünk is el kell merülni ebben a világban. Nézzük tehát a ***Samba Domain Controller*** szerepét.

Elméleti alapok

Mielőtt mélyebbre ásnánk magunkat, néhány alapvető fogalommal mindenképpen meg kell ismerkednünk. Szeretném továbbá, ha az olvasók az alábbi szemléletben járnának el a rendszer telepítése és használata során:
„...Az teljesen helyénvaló, hogy hibákat kövessünk el, abban az esetben, ha azokat a hibákat ott és akkor követjük el, ahol ennek a helye van. Semmiképpen nincs helye hibák elkövetésének ott, ahol ezzel mások munkáját akadályozhatjuk, vagy ezzel másoknak kárt okozhatunk.
Amennyiben kísérletezni szeretnénk egy rendszeren, tanulni szeretnénk annak szolgáltatásait, azt mindenképpen egy olyan tesztkörnyezetben tegyük, ahol ezzel mások munkáját nem hátráltatjuk."
(A ➲ samba.org útmutatása alapján.)

Miért is kell nekünk a tartomány, mi előnyünk származik belőle?

A válasz egy mondatban összefoglalható, ez pedig a ***Single Sign On***, avagy az egyszerű bejelentkezés. *Windows* rendszerekben azokra a munkaállomásokra, amelyek a tartomány tagjai a felhasználó mindenhol ugyanazokkal a felhasználói paraméterekkel léphet be és munkájához ugyanazt a munkakörnyezetet és erőforrásokat használhatja.
Nézzük melyek azok a szolgáltatások, amelyeket a ***Samba 3***-as verziója ezen a téren kínál nekünk. Először is úgynevezett ***trust relationship***-et, vagyis bizalmi kapcsolatot tudunk létesíteni az egyes *Windows NT4*-es tartományokkal, valamint ezek tagjaival. Ez azért hasznos nekünk, mert két olyan tartomány, amelyek bizalmi kapcsolatban állnak, különösebb problémák nélkül tudnak erőforrásokat megosztani egymással. Csak zárójelben jegyezném meg, hogy a *Windows Server* család *Small Business* kiadásai nem tudnak más rendszerekkel ***bizalmi viszonyt*** kialakítani. Ez egy beépített korlátozás, mivel ezeket a szoftvereket kisvállalati környezetre tervezték és a *Microsoft* filozófiája szerint egy kisvállalati környezetben maximum egy kiszolgálót használnak. Ez van sajnos. Nézzük meg, mit veszünk, mielőtt a bomba vételből bomba meglepetés lesz.
Egy másik kényelmi szolgáltatása a ***Samba 3***-as rendszernek, hogy a tartományban lévő munkaállomásokról a beépített *Windows NT* segédprogramokkal tudjuk kezelni többek között a felhasználókat. Fontos, hogy csak a *Windows NT 4*-es verziójához járó segédprogramok felelnek meg a feladatra, a *Windows 2000*, *XP* és *2003 Server* nem. A programok megtalálhatóak az *SVRTOOLS* csomagban, amely letölthető a Microsoft weblapjáról, vagy lemásolható bármely *Windows NT 4 Server CD*-ről. Szintén új szolgáltatása a ***Samba 3***-asnak, hogy a felhasználói adatbázist tartalmazó háttérszolgáltatás cserélhető, tehát bármelyik olyan úgynevezett ***backendet*** használhatjuk, amelyik a céljainknak megfelel és a *Samba* is támogatja. Így lehetséges például *LDAP*, vagy *MySQL backend* használata. Az utolsó nagy újítás, amit kiemelnék az a ***Unicode*** támogatás, amelyre a legújabb *Windows* kliensek esetén már szükségünk lehet.

Mire nem használható a Samba

A *Samba* nem képes jelenleg *Windows* kiszolgáló ***kisegítő tartományvezérlőjeként (Backup Domain Controller)*** működni és *Windowst* futtató kiszolgálóhoz sem tudunk *Samba BDC*-t illeszteni. Ez van, tervezzünk ügyesen és éljünk ezzel együtt, amíg megoldás nem születik rá.
A *Samba* annak ellenére, hogy rendelkezik némi támogatással a *Windows 2000* tartományvezérléshez, a *Samba* nem tudja helyettesíteni a *Windows 2000* tartományvezérlők minden funkcióját. A dokumentáció szerint azonban az illetékesek dolgoznak a problémán és várhatóan a *Samba 3* egy későbbi kiadása során, vagy az azt követő verzióban már szerepelni fog ez a szolgáltatás is.

A „titokzatos" tartományvezérlők

A tartományvezérlőknek három típusát különböztetjük meg, a *PDC-t*, vagyis ***elsődleges tartományvezérlőt*** (***Primary Domain Contoroller***), a ***kisegítő tartományvezérlőt (Backup Domain Controller; BDC)***, valamint a *Windows 2000 Server* és *Windows Server 2003* rendszerek esetén az *ADS Domain Controller*, amely egy *Active Directory* támogatással rendelkező tartományvezérlő.
A *PDC* szerepe kiemelt egy tartomány életében, hiszen a rendszerben az azonosítási feladatok ellátása ennek a gépnek a feladata. Él ugyanakkor egy tévhit a köztudatban, miszerint az elsődleges tartományvezérlőt érdemes a hálózatban a legizmosabb gépre telepíteni. Ez nem feltétlenül igaz sőt, nagy hálózatokban érdemes a szolgáltatásokat több különálló tartománybeli kiszolgálóra szétosztani. Ez erőforrás elosztás és skálázhatóság tekintetében is egy jó ötletnek tűnik.
A *BDC* szerepe egy tartomány életében annyi, hogy állandóan másolja a *PDC* tartomány adatbázisát és amennyiben az elsődleges tartományvezérlő elérhetetlenné válna, úgy átveszi annak szerepét. Látszik, hogy tartalék tartományvezérlő megléte nem szükséges, de nagy hálózatok és kritikus környezetek esetén hasznos lehet.
A *Windows 2000 Server* és utódai esetén a tartományvezérlés egy kicsit megváltozott. Megszűnt az elsődleges és tartalék szerep, helyette a tartományvezérlők egy fa struktúrába vannak rendezve. Egy tartományvezérlő felelős az ő részfájában található gépek kiszolgálásáért, valamint tudni kell azt is, hogy ezen szolgáltatásokat a fában feljebb álló kiszolgálók felüldefiniálhatják. *Samba 3* esetén ez a működés *LDAP backend* használatával hasonló módon megvalósítható.
Windows NT 4-es környezetben egy kiszolgáló négy szerepet tölthet be. Ezek közül kettő a már említett elsődleges és tartalék tartományvezérlő, míg a további két szerep a ***stand-alone***, tehát egyedülálló szerver, valamint ***domain member***, tehát tartománytag kiszolgáló. Ennek a szerepnek az eldöntéséről a kiszolgáló telepítésekor kell dönteni. Amennyiben valamelyik tartományvezérlőt választjuk típusnak, úgy lehetőség van arra, hogy menet közben áttérjük a másik típusú tartományvezérlő szerepre, amennyiben erre a hálózat felépítése lehetőséget ad. Egyéb változtatások elvégzésére egyetlen út van, a kiszolgáló újratelepítése.
Amennyiben ***domain member*** kiszolgálót telepítünk, úgy a ***felhasználó-hitelesítő adatbázisunkat (Security Account Manager; SAM)*** az elsődleges tartományvezérlő fogja biztosítani. Amennyiben ***stand-alone*** kiszolgálót telepítünk, akkor a kiszolgáló ezeket az adatokat lokálisan fogja tárolni, így amennyiben változtatni szeretnénk, úgy a változásokat ezen a kiszolgálón kell átvezetni.
Már említettem, hogy a *Samba* egyelőre csak kísérleti stádiumban van ami a *Windows 2000 Active Directory* tartományvezérlést illeti, ugyanakkor a *Samba 3*-as kiszolgálók teljes értékű tartományi tagkiszolgálóként (***domain member)*** képesek egy ilyen környezetben is működni.
A sok-sok elmélet után nézzük miként is lehet megszerzett tudásunkat a gyakorlatban is alkalmazni.

A tartományvezérlés előkészítése

A *Windows* kliensek kétféle módon tudnak egymással hálózati kapcsolatba kerülni. Az első, hogy egy munkacsoport, úgynevezett ***munkacsoport (workgroup)*** tagjaivá tesszük őket, a másik, hogy a tartomány tagjaivá válnak. Előbbi a gyakorlatban nem jelent többet annál, minthogy van egy munkaállomásunk, amelyik más munkaállomásokhoz hozzáférhet, ha azon a megszólított munkaállomáson ez engedélyezve van. Amennyiben a kliensünket a tartomány tagjává tesszük, úgy az biztonságosan és szabadon tud kommunikálni a tartomány többi gépével.
Tartomány használata esetén a tartományi adatbázisba nem csak a felhasználók kerülnek bejegyzésre, hanem minden olyan kliens is, amelyiket hozzáadtuk a tartományhoz, így a biztonságosan használható klienseket is tároljuk.
Nézzük milyen feltételeknek kell egy *Samba 3*-as kiszolgálónak megfelelni, hogy az egy *Windows NT 4*-es tartományvezérlő szolgáltatásait biztosítsa a kliensek felé:

- Össze kell állítanunk egy működő *TCP/IP*-s *Windows* hálózatot. Ez úgy gondolom elég triviális dolog.
- A *Samba* beállításainál a megfelelő biztonsági szabályt kell választani, amely a `security = user` változó-paraméter párral lehetséges.
- A hálózatnak kell rendelkeznie egy jó beállított névfeloldással.
- A klienseket tartományi belépésre kell beállítani
- Ajánlatos a felhasználói profilokat úgynevezett ***barangoló profilként (roaming profile)*** létrehozni, így a felhasználó a használt munkaállomástól függetlenül minden gépen azonos környezetben dolgozhat.
- A kliensek felé biztosítani kell a *Netlogon* szolgáltatást, amelyet a konfigurációs állományban lehet beállítani.
- Javasolt, hogy a kiszolgáló vegye át a rendszerben az úgynevezett *Master Browser* szerepet, ugyanis ezzel hatékonyabbá lehet tenni a működést.

Nézzünk egy példát a beállításokra:

```
[global]
netbios name = MYSERVER
workgroup = MYDOMAIN
passdb backend = tdbsam
os level = 33
preferred master = yes
domain master = yes
local master = yes
security = user
domain logons = yes
```

```
logon path = \\%N\profiles\%U
logon drive = H:
logon home = \\homeserver\%U\winprofile
logon script = logon.cmd

[netlogon]
path = /var/lib/samba/netlogon
read only = yes
write list = ntadmin

[profiles]
path = /var/lib/samba/profiles
read only = no
create mask = 0600
directory mask = 0700
```

A passdb backend tartalmazza az összes felhasználóifiók és csoport információt. A lehetséges értékek: `smbpasswd`, `tdbsam`, és `ldapsam`. Amennyiben *BDC*-t is használni szeretnénk a rendszerben, úgy az egyetlen értelmes választási lehetőség az `ldapsam`, mivel az `smbpasswd` és `tdb` adatbázisok lokálisak, így nem lehet őket a hálózatban elterjeszteni.

A tartományvezérlő paraméterei

A tartományvezérlő lehetséges paraméterei: `os level`, `preferred master`, `domain master`, `security`, `encrypt passwords`, és `domain logons`.
Ezek természetesen központi szerepet játszanak a tartományvezérlő tulajdonságainak kialakításakor. Az `os level` változó értékét mindenképpen állítsuk 32-nél nagyobbra, hogy a megfelelő prioritást megadjuk a kiszolgálónak.
A `preferred master` változót szintén állítsuk igaz értékre, mivel ezzel tudjuk aktiválni azt, hogy a tartományban ez a gép legyen a `master browser`. A `domain master` változónak adjuk a `yes` értéket, amennyiben a kiszolgáló egy elsődleges tartományvezérlő, amennyiben azonban egy tartalék tartományvezérlőt állítunk be, úgy logikusan a `no` értéket adjuk a változónak.
Tartományvezérlő beállítása esetén a `security` változó a `user` értéket kapja, ami azt jelenti, hogy a megosztásokhoz való hozzáférés alkalmával a kapcsolatot kezdeményező kliens egy felhasználói azonosításon is átesik, a megadott felhasználónak léteznie kell a kiszolgáló adatbázisában.
A tartományvezérlő helyes működéséhez a klienseknek és a kiszolgálónak is támogatnia kell a *Microsoft* titkosított jelszókezelését. Ehhez a kiszolgálón az `encrypt passwords` változót állítsuk `yes` értékre.
Végül, de nem utolsó sorban ahhoz, hogy a kliensek igénybe vehessék a *Network Logon* szolgáltatást, a `domain logons` paraméternek is adjunk `yes` értéket. A *Network Logon* szolgáltatáshoz kapcsolódni kell továbbá egy `netlogon` kiosztás meglétének is, ahol a rendszerbe belépő kliensek megtalálják a *login szkriptet*, tehát azt a futtatható állományt, amit minden tartományba való belépéskor le kell futtatniuk.

Környezeti változók

A lehetséges környezeti változók a következők: `logon path`, `logon home`, `logon drive`, és `logon script`.

Ezek segítségével olyan működési környezetet tudunk kialakítani az ügyfelek és a felhasználók számára, amely a legjobban illeszkedik az elvárt igényekhez.
A `logon path` változó értékének a felhasználói profil helyét kell megadni. Az elérési út megadásánál használhatunk reguláris kifejezéseket is. A `logon drive` annak a meghajtónak a betűjelét mutatja, amely hálózati meghajtón keresztül a klienseken a `logon home` változó értékének megadott könyvtárat érhetik el. Ez a könyvtár lesz a felhasználó privát hálózati meghajtója. Amennyiben a rendszerben vannak olyan felhasználók is, akik notebookokról érik el a hálózatot, akkor náluk az újabb *Windows* rendszerekben érdemes a kapcsolat nélküli elérhetőséget erre a hálózati meghajtóra beállítani.
A `logon script` az az állományt mutatja, amelyik a felhasználók belépésekor lefut minden gépen.

A NETLOGON megosztás

A `NETLOGON` megosztás egy alapértelmezett megosztás minden Microsoft tartományvezérlőn. Ez a mappa tárolja a *logon scriptet*, a házirend állományokat (*NTConfig.POL* állomány, amelyben az ügyfél és a felhasználó hozzáférését tudjuk korlátozni az egyes rendszerbeállításokhoz), valamint minden olyan állományt,amelynek a belépési procedúra során feldolgozásra kell kerülnie.

A PROFILE megosztás

Ebben a megosztásban tárolja a *Windows* a felhasználók profiljait. Amikor egy felhasználó belép a rendszerbe, akkor a kliens gép lekéri a felhasználói profilját, betölti a szükséges környezetet és átadja munkára a felhasználónak. Kilépéskor ugyanez megtörténik ellenkező irányban, tehát a felhasználó minden megváltozott beállítása elmentésre kerül. Érdemes a *UNIX* jogosultságokat úgy beállítani, hogy az adott felhasználói profilhoz csak a tulajdonosa férhessen hozzá. Ez növelheti a rendszer biztonságát.
Ezzel cikkem végére is értem. Mostanra sikerült egy tartományvezérlőt beállítani és elsajátíthattuk a kapcsolódó ismeretek alapjait. Annyit azért érdemes megjegyezni, hogy egy *Samba* kiszolgáló jelenleg még nem képes minden *Windows* szolgáltatás teljes kiváltására és valószínűleg mindig egy lépéssel a *Microsoft* mögött fog járni, de már ez az egy lépés hátrány olyan mértékűvé csökkent, amikor érdemes elgondolkozni azon, hogy a két rendszer tudásbeli és szemléletbeli különbsége közötti arány van-e olyan mértékű, amikor még érdemes lehet egy *Windows* kiszolgálót üzembe állítani.
Aki komolyan szeretne a témával foglalkozni, az kövesse továbbra is figyelemmel a *Linuxvilág magazint*, mert ez a téma ezen a cikksorozaton kívül is egész biztosan felmerül majd. Érdemes továbbá a nyílt forráskódú rendszerekkel foglalkozó portálokat is sűrűn látogatni, mert minden nap újabb és újabb megoldások kerülnek fel a hálózatra.
Végezetül mindenkit arra buzdítok, hogy nyugodtan játsszon a rendszerrel, próbálgassa a beállításokat – persze ha lehet, akkor a *Samba* fejlesztőinek ajánlásait betartva.

Illés Viktor

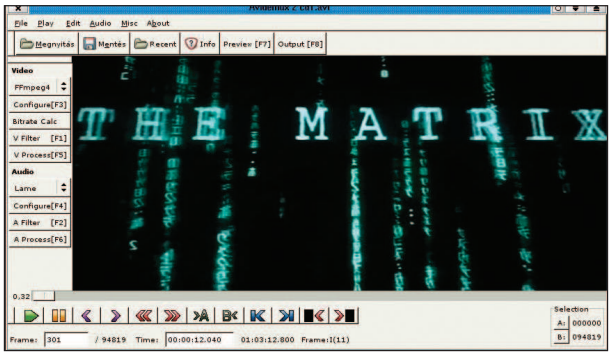
FreeBSD – a szomszéd vár (5. rész)

Az előző részből már csak néhány „apróság” maradt ki, amelyek még szükségesek egy átlagos munkaállomáson: a nyomtatás, a hálózat beállítása, illetve CD és DVD írás.

A legnagyobb kihívást a nyomtatók beállítása jelenti, ezek között van néhány amelyeket a *FreeBSD* egyik meghajtója vagy segédprogramja sem támogat. A hálózat már egyszerűbb – bár sokrétűbb feladat – mivel a *FreeBSD* lételeme a hálózat, így nagyon kevés eszközt nem támogat, amelynek köze van bármilyen hálózati kapcsolathoz. Végül a *CD* és *DVD* készítéséről írok, amely szintén nem okoz nehézséget, mivel része az alaprendszernek egy *CD*-író program.

Nyomtatás

A nyomtatás lehetősége már hozzátartozik a legtöbb számítógéppel végzett tevékenységhez, szinte minden felhasználói program támogatja a nyomtatás lehetőségét, legyen a nyomtatás tárgya egy levelezőprogram esetén egy levél, egy böngészőprogram ablakának tartalma, akár egy képszerkesztő programban egy retusált fénykép.
Legegyszerűbb dolgunk akkor van, ha a nyomtatónk képes értelmezni a *PostScript (PS)* állományokat, mivel a legtöbb program ebben a formátumban próbálkozik meg a nyomtatással. Azt is mondhatnánk, hogy *UNIX* alatt a programok kivétel nélkül szimpla szövegként, vagy *PostScript* nyelven próbálnak nyomtatni.
Sajnos a legtöbb manapság kapható nyomtató nem érti a PostScript nyelvet, és pár éve már a szimpla szöveges (úgynevezett *DOS kompatibilis*) nyomtatást sem. A nyomtatók egyszerűen csak így lehettek annyira olcsók, hogy szinte bárki képes legyen megvenni egyet (értem ez alatt a *PostScript* és a szöveges implementáció teljes hiányát). Nem kell azonban elkeseredünk, ha csak egy olcsó lézernyomtatónk van, hiszen ezek leginkább cégeknél vannak használatban, és ezért a legtöbbet támogatja a *GhostScript* programcsomag, amely a `ports` adatbázisból feltelepíthető, és a `print` kategóriában a `ghostscript-gnu` alatt található meg. Sajnos nem triviális dolog beállítani a megfelelő szűrőket, amelyeket az `lpd` majd kezelni fog, ezért rövidítsünk a problémán: telepítsük fel az `apsfilter` csomagot (*/usr/ports/print/apsfilter*). Ennek segítségével könnyedén be tudjuk állítani a rendszert a nyomtatáshoz. Egyetlen megkötése van a program használatának: úgynevezett *Cardware* kategóriába tartozik, vagyis nem ingyenes a használata. Nem kell megijedni, csak küldenünk kell egy képeslapot a szerző címére, ez maximum párszáz forintunkba kerül, ennyit megér a használata. Az elején egy szándéknyilatkozatot kell elfogadni, amelyben megígérjük, hogy küldünk majd egy lapot. A beállítás a



1. ábra

```
$ /usr/local/share/apsfilter/SETUP
```

parancs kiadása után indul el. Néhány ismertető képernyőszöveg után eljutunk a program főmenüjéig:

```
(D)  Available Device Drivers in your gs binary
(R)  Read Ghostscript driver documentation
     ➥(devices.txt)

(1)  Printer Driver Selection      [pcl3/unspec]
(2)  Interface Setup               [parallel]
(3)  Paper Format                  [a4]
(4)  Printing Quality              [high]
(5)  Color Mode                    [full]
(6)  Print Resolution in “dots per inch”
     ➥[1200x1200]
(7)  Default Printing Method       [auto]

(T)  Print Test Page
(V)  View performance log (times of print attempts)

(A)  Abort installation (don’t do anything)
(I)  ==> Install printer with values shown above -
         ➥repeat this
```

```
        step for installing multiple printers
(Q)  ==> Finish installation
```

A beállítás nem bonyolult, a számozott lépéseken kell áthaladnunk, kiválasztjuk a nyomtatónk kategóriáját (milyen meghajtóprogram-csoport kezelje), majd a nyomtató pontos típusát. Ezek után kiválasztjuk a csatoló felületet, vagyis azt, hogy milyen kábellel kötöttük össze a gépünkkel (párhuzamos, ***USB***, soros), illetve milyen eszköznévvel éri majd el a program (***/dev/lpt0***, ***/dev/ulpt0***). A papírméret, a minőség, a színek jellege, illetve a felbontás triviális válaszokat igényel. A beállítások végeztével tudunk próbanyomatot kérni a `T` megnyomásával, ha ez sikeres volt, akkor az `I` lenyomásával hozzáadhatjuk a rendszernyomtatókhoz az általunk megadott néven. Mindenképpen készítsünk egy általános `lp` nevű nyomtatósort, mivel ezt keresi a legtöbb program. Ellenben érdemes több kombinációt elkészíteni (például `lp-mono-300-draft`, `lp-color-1200-photo`, ...), így a megfelelő nyomtatósor kiválasztásával kiválasztjuk a nyomtatás jellemzőit is. A `Q` megnyomásával elkészül a */etc/printcap* állomány, ezzel végeztünk is a nyomtató beállításával, bármilyen programból képesek leszünk nyomtatni.

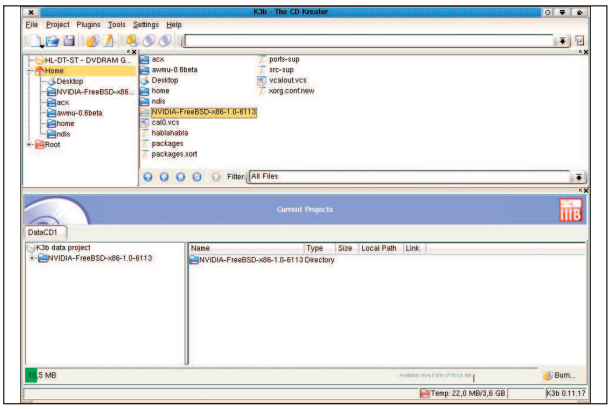
Hálózat

Manapság hálózati hozzáférés nélkül nehéz boldogulni, gondoljunk itt vállalati vagy otthoni munkaállomásra. Legkönnyebb dolgunk egy vállalati hozzáférés kialakításánál van, mivel ott gyakorlatilag csak ***UTP*** kábellel kialakított ***LAN*** található, egyszerűen csak be kell állítanunk ***DHCP***-t vagy fix ***IP*** címet, és a megfelelő átjárót (esetleg a ***proxy*** kiszolgáló címét). Ezen paraméterek megszerzéséhez a vállalat rendszergazdáját kell megkörnyékeznünk.
Otthoni hálózati hozzáférés nagyon sokrétű, sokkal nehezebb dolgunk van ezek beállításával: a modem, az ***ADSL***, a ***GPRS*** vagy a ***WiFi*** mind egy-egy megoldás a sok közül. Mindegyik módszer más-más alrendszert igényel az internetre való kapcsolódás folyamán.

Modem és ISDN

10 éve szinte csak modemmel lehetett hozzáférni a világhálóhoz, így ez a legősibb és egyben legstabilabban implementált kapcsolódási mód. Legalábbis ez sokáig igaz volt, amíg a költségcsökkentés okán a gyártók el nem kezdtek a modemeken is spórolni. (Tizenegy éve egy 28800bps külső modem számított a csúcsnak, amelynek ára a mai minimálbér, vagy ha úgy tetszik az akkori közalkalmazotti átlagkereset majd kétszerese volt. Máig őrzöm, mert még anyag van benne...). Ennek megfelelően érthető a spórolás oka, viszont a módszer nagyon sok ablakmentes felhasználónak okozott szenvedést. A modemet vezérlő szoftvert átmozgatták a számítógép oldalára, így az operációs rendszerre bízták annak működtetését meghajtóprogramot pedig csak ***Windows*** alá írtak. Sajnos az utóbbi öt évben kapható modemek – kevés kivételtől eltekintve – mind ilyen szoftver-modemek, amelyek ***Linux*** általi támogatása is nagyon lassan fejlődött ki, ***FreeBSD*** alatt viszont szinte csak a hardver-modemek a támogatottak.
Ennek megfelelően a rendszermag által írt üzenetek között találnunk kell a

```
sio0 at 0x3f8-0x3ff irq 4 on isa
sio0: type 16550A
```



2. ábra

hasonló sorokat, és ekkor a */etc/ppp/ppp.conf* állományt megfelelő módon szerkesztve képesek leszünk a szolgáltatónk által biztosított internet elérés használatára. Ha ilyet nem találunk, akkor nagy eséllyel – sajnos – nem fogunk hálózathoz csatlakozni modemmel.
ISDN modemek nagy része támogatva van, meg kell keresnünk a soros kommunikációt lehetővé tévő eszköz nevét, s azt kell használnunk.
A ***ppp.conf*** szerkesztése nem olyan nehéz, mint először látszik. A legrövidebb beállítás alább látható, a vastagon szedett szöveget kell csak szerkeszteni, illetve több szolgáltató esetén az elkülönülő blokkot kell még egy példányban (más elnevezéssel) lemásolnunk:

```
default:
   set log Phase Chat LCP IPCP CCP tun command
   ident user-ppp VERSION (built COMPILATIONDATE)
   set device /dev/cuaa0
   set speed 115200
   set dial “ABORT BUSY ABORT NO\\sCARRIER
   ➥TIMEOUT 5 \
           \”\” AT OK-AT-OK ATE1Q0 OK \\dATDT\\T
           ➥TIMEOUT 40 CONNECT”
   set timeout 180
   enable dns

szolgaltato:
   set phone “+3651-xxx-xxx”
   set authname username
   set authkey password
   set login “TIMEOUT 10 \”\” \”\” gin:–gin: \\U
   ➥word: \\P col: ppp”
   set timeout 300
   set ifaddr 0.0.0.0 255.255.255.255
   ➥255.255.255.255 0.0.0.0
   add default HISADDR
```

A tárcsázás nem okoz nehézséget, be kell írnunk (root joggal) a parancssorba, hogy `ppp`, majd kiadni a `dial szolgaltato` parancsot.

ADSL

Nagyfokú különbség nem fogad minket, ha ***ADSL***-t szeretnénk használni, egyszerűen a megfelelő hálókártyába

dugjuk az *ADSL* „modemet” a kapott kábellel, majd létrehozunk egy *ppp.conf* állományt a következők szerint (a vastagon szedett szöveget szerkesztjük):

```
default:
  set log Phase tun command # you can add more
  ➥detailed logging if you wish
  set ifaddr 10.0.0.1/0 10.0.0.2/0

szolgaltato:
  set device PPPoE:xl1
  set authname username
  set authkey password
  set dial
  set login
  add default HISADDR
```

Az `xl1` helyett a megfelelő hálózati csatoló megnevezését kell írnunk. A tárcsázás azonos módon történik, mint modem esetén.

GPRS

Dinamikus és gyorsan fejlődő ága a hálózatoknak a mobil végpontok használata. Ekkor egy laptoppal és egy mobiltelefonnal a világ végéről is tudunk kapcsolódni a világhálóra. Több elterjedt módja is van a mobiltelefon számítógéphez való kapcsolásának: soros kábel, ***USB*** kábel, ***IrDA*** vagy ***BlueTooth***.
A legkényelmesebb megoldás ***FreeBSD*** szempontjából a kábeles kapcsolat, amelyet soros portra kell kötnünk. A ***GPRS*** képes telefonok igen nagy része képes modemként működni, így a `ppp` programot tudjuk kapcsolódásra használni.
A *ppp.conf* sem lesz összetettebb, mint modemes kapcsolat esetén:

```
default:
 set log Phase Chat LCP IPCP CCP tun command
 ident user-ppp VERSION (built COMPILATIONDATE)
 set redial 2.2 300
 set reconnect 2.2 300
 set device /dev/cuaa0
 set speed 115200
 set timeout 180
 enable dns

szolgaltato:
 enable force-scripts
 disable ipv6cp
 set lcp-echo-failure 0
 set authname username
 set authkey password
 set phone “*99***1#”

 set dial “ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \
          \”\” AT OK-AT-OK ATE1Q0 OK \\dATD\\T
          ➥TIMEOUT 40 CONNECT”
 set logout “ABORT BUSY ABORT ERROR TIMEOUT 30 \
           \”\” +++ATH OK-ATH-OK AT+CGATT=0 OK”
 set login
 set timeout 300
 enable dns
 resolv rewrite

 set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.0
 ➥0.0.0.0
 add default HISADDR
```

USB kábelek közül a legtöbbön egy kis lapka ***USB***-soros átalakítást csinál, amelyet az operációs rendszer visszaalakít soros kommunikációra a megfelelő meghajtóprogrammal. A gyakori lapkákat támogatja a ***FreeBSD***, de azért ne bízzuk a véletlenre – próbáljuk ki megvásárlás előtt a kábelt.
Infravörös átvitel sokáig divatos volt a mobil kommunikációban, nagyon sok gyártó készít asztali gépekhez ***IrDA*** csatlakozót, ezek egy része működni is fog ***FreeBSD*** alatt: egy soros port fog létrejönni, ezt tudjuk megadni a *ppp.conf* állományban és egyszerűen csak tárcsáznunk kell.
Leggyakoribb és legkényelmesebb megoldás a ***BlueTooth*** kapcsolódás, szinte az összes 1-2 éves telefon ismeri a kékfogat, mint adatátviteli módszert. A számítógép 10-15 méteres sugarában bárhol lehet a telefon, nincs egy másfél méteres kábelhez vagy infravörös látótávolsághoz kötve. Praktikus is, mivel egy ***USB BlueTooth*** végpont annyiba kerül, mint egy adatkábel, viszont szabadon válthatunk telefont, a ***BlueTooth*** azzal is működni fog. ***FreeBSD*** alatti beállítása és használata önmagában egy egész cikket megtöltene, ezért most csak a lényegi részére szorítkozok (a többi olvasható – egyelőre – angol nyelven a ***FreeBSD*** kézikönyvben).
Egy `rfcomm` nevű alrendszerre lesz szükségünk, amely a széleskörűen használható, szélessávú ***BT*** adatcsatornából kihasít egy részt virtuális soros kommunikáció céljára. Ehhez egyedül a telefon ***BT MAC*** címére lesz szükségünk:

```
$ rfcomm_pppd -a 00:0E:07:62:57:28 -d -c -C dun -l
➥szolgaltato
```

A program futása során létrehoz egy ilyen virtuális csatornát, és elindítja rajta a `ppp` démont a *ppp.conf* állományban leírtak alapján (`szolgaltato`).

WLAN és WiFi

Ha rendelkezünk egy megfelelően konfigurált ***802.11g*** vagy ***802.11b WLAN*** kártyával, akkor előttünk a lehetőség a vezeték nélküli szélessávú hálózatok használatára (akár 54Mbps elérésére is). A legtöbb ***WiFi*** kártya (legyen az ***PCI*** vagy ***PCMCIA***) támogatott ***FreeBSD*** alatt, ha nem natív meghajtóprogrammal, akkor az `NdisWrapper` segít ebben, amelyet az előző részben már említettem.
Sikeresen beállított ***WiFi*** kártya esetén az `ifonfig wi0` parancs hatására az alábbit láthatjuk:

```
wi0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,
➥MULTICAST> mtu 1500
        inet6 fe80::202:2dff:fe2d:c938%wi0
        ➥prefixlen 64 scopeid 0x7
        inet 0.0.0.0 netmask 0xff000000 broadcast
        ➥255.255.255.255
        ether 00:09:2d:2d:c9:50
        media: IEEE 802.11 Wireless Ethernet
        ➥autoselect (DS/2Mbps)
        status: no carrier
        ssid “”
```

```
        stationname "FreeBSD Wireless node"
        channel 10 authmode OPEN powersavemode OFF
        ➥powersavesleep 100
        wepmode OFF weptxkey 1
```

A beüzemeléshez mindenképpen meg kell adnunk egy SSID azonosítót, például

```
$ ifconfig wi0 ssid 'azonosito'
```

s ezek után már a

```
    status: associated
    ssid "azonosito"
```

szöveget kell látnunk majd, ha az adott nevű hálózat vételi körzeten belül van. Ezzel hozzá tudunk kapcsolódni egy nyilvános vezeték nélküli hálózathoz, a legtöbb esetben azonban ***WEP*** titkosítással vannak a lefedett körzetek védve. A ***WEP*** megadásához szintén az `ifconfig` parancsot tudjuk használni:

```
$ ifconfig wi0 wepmode on wepkey 0x0123456789ABCDEF
```

Az ***IP*** cím és a többi információ megadása – beállítása azonos módon történik egy átlagos hálózati kártya esetén megszokott módszerrel.

Levelezés

Alapvetően minden szükséges program megvan az alaprendszerben, ha levelezni támad kedvünk, hiszen egy ***Sendmail*** programot használhatjuk levelek fogadására és küldésére. Alapbeállításképp a ***sendmail*** nem fogad leveleket a gépünk IP címén, csak a `127.0.0.1/localhost` címen, de ezen változtathatunk a */etc/rc.conf* szerkesztésével

```
sendmail_enable="YES"
```

és ezáltal már (megfelelő beállítások után) fogad kívülről is leveleket a program. Ha nem használjuk levélküldésre a gépünket – például a szolgáltató SMTP szerverét használjuk, akkor a */etc/rc.conf* állományba a

```
sendmail_enable="NONE"
```

sort írjuk be. Ezáltal a ***sendmail*** nem fog elindulni, így ennyivel több erőforrás marad a többi programunk számára. Levelező programok közül szinte az összeset megtaláljuk a `ports` adatbázisban, legyen az a `pine` vagy a `mutt`; illetve grafikus felületen az ***Evolution*** vagy a ***Kmail***.

CD írás

A ***FreeBSD*** olyannyira támogatja az optikai tárolóeszközök (***CD***/***DVD***) használatát, hogy az alaprendszere már (az 5.0, illetve a 4.8 verziók óta) tartalmaz ***CD***-író programot: a `burncd`-t, amelyet a `/usr/sbin/burncd` helyen találunk meg. Sajnos – a többi íróprogramhoz hasonlóan – nem alkalmas arra, hogy egyszerű felhasználóként ***CD***-t írhasson bárki is, mivel erre nincs kitaposott megoldás (esetleg a `sudo` tűnhet még jó ötletnek).
A `burncd` használata önmagában kissé nehézkes, mivel csak ***ATAPI*** írókat támogat, illetve ***nincs*** az alaprendszerben olyan program, amely elkészíti az ***ISO9660*** fájlrendszert, s erre szükségünk lesz egy összetettebb egyedi ***CD*** vagy ***DVD*** írásához, ezért érdemes lesz feltelepítenünk az mkisofs programot (a `cdrtools ports` csomag része). Ezáltal tudunk zenét ***CD***-re égetni, akár ***CD***-t másolni, de például mentésre és adatok felírására is használható ez a program...
Természetesen a `ports` adatbázisban találunk egypár segédprogramot, amelyek jobban és kényelmesebben használhatók a ***CD***/***DVD*** készítésekor.
A legfontosabbat már említettem: az `mkisofs` nélkülözhetetlen eszköz a ***CD***/***DVD*** fájlrendszerének elkészítéséhez. Használata teljesen azonosan történik, mint ***Linux*** esetén, a paraméterezése is ugyanaz.
Sokkal jobban járunk, ha a `burncd` helyett a `cdrecord` programot használjuk, amelyet jó esetben az `mkisofs` programmal együtt már feltelepítettünk, ugyanis a `cdrtools` csomag része. A `cdrecord` támogatja a ***SCSI*** és az ***USB*** íróeszközöket is.
Érdekességképpen az ***ATAPI CD*** meghajtók esetén a zenét tartalmazó ***CD***-k egyszerűen másolhatók, mivel létrejön a */dev/acd0* eszközön kívül minden sávhoz tartozni fog egy */dev/acd0tnn* eszköz, amely az adott sávot tartalmazza. Ezt a másolást legegyszerűbben a

```
$ dd if=/dev/acd0t01 of=track01.cdr bs=2352
$ dd if=/dev/acd0t02 of=track02.cdr bs=2352
```

paranccsal tudjuk megtenni, természetesen a megfelelő számú sávval és megfelelő nevű eszközzel. A kapott ***.cdr*** állományokat fel tudjuk írni egy üres ***CD***-re a következő paranccsal:

```
$ burncd -f /dev/acd0 audio track1.cdr track2.cdr
➥... fixate
```

Természetesen a `burncd` segítségével képesek vagyunk CD lemezre írni az aktuális mentést, ehhez a

```
$ burncd -f /dev/acd0 -s 12 data archive.tar.gz
➥fixate
```

parancsot kell kiadni, amely úgy írja a ***CD*** lemezre az adatot, hogy az különösebb fájlrendszerként funkcionálna.
A probléma mindössze annyi, hogy nem tudjuk felcsatolni a fájlrendszerbe, hanem a

```
$ tar xzvf /dev/acd1
```

parancs magáról a ***CD*** eszközről binárisan olvasva tömöríti ki a ráírt állomány tartalmát. Nos, ezt a kis nehézséget tudjuk az `mkisofs` használatával áthidalni...
Mivel a téma igazából a munkaállomások beállítása, azért feltelepíthetünk akár grafikus felületű CD író programot, mint például az ***X-CD-Roast*** (`/usr/ports/sysutils/xcdroast`) vagy a ***K3b*** (`/usr/ports/sysutils/k3b`). Ezekkel kényelmesebbé válik az írás, viszont nehézséget okozhat, hogy az eszközhöz root jogok kellenek, így vagy root jogokkal használjuk a grafikus felületet, vagy – például – a `kdesu` programmal oldjuk meg a rendszergazdai jogokkal való futtatást...

DVD írás

DVD írásához már más eszközök szükségesek, mint például a `growisofs`, amely a `dvd+rw-tools` (`/usr/ports/sysutils/dvd+rw-tools`) csomag része. A ***DVD*** íráshoz nem szükséges használnunk az `mkisofs` programot,

de telepítve rendelkeznünk kell vele, mert a program meghívja azt. Egy egyszerű adat *DVD* írásához a

```
$ growisofs -dvd-compat -Z /dev/cd0 -J -R
➥/home/usernev/konyvtar/
```

parancsot kell kiadni, és ezzel a megadott könyvtár tartalma a *DVD*-re íródik. Az újraírható *DVD* típusok (*DVD±RW*) írása hasonlóképpen történik, ezekhez hozzá tudunk írni (`-M` opció), illetve törölni is tudjuk a tartalmát (`-Z` opció):

```
$ growisofs -Z /dev/cd0 -J -R
➥/home/usernev/konyvtar/
$ growisofs -M /dev/cd0 -J -R
➥/home/usernev/konyvtar/
```

A *DVD+RW* lemezeket azonban először meg kell formázni, amelyhez a

```
$ dvd+rw-format /dev/cd0
```

parancsot kell használnunk. Többször is megformázhatunk egy-egy *DVD+RW* lemezt, bár ennek nincs sok értelme, akkor próbáljuk ki, ha nagyon nem tudjuk már kezelni a lemezt. Képesek vagyunk DVD-RAM lemezeket is használni, személy szerint mentéshez ajánlom ezeket, mert a lemezeket szokásos módon megformázzuk

```
$ newfs -U -L "cimke1" -m 1 /dev/cd0
```

majd - szintén szokásos módon – felcsatoljuk a megfelelő helyre:

```
$ mount /dev/cd0 /mnt
```

Ezek után úgy tudjuk használni, mint egy 4.5G-s nagyon gyors hajlékonylemezt (vagy lassú merevlemezt). Bármilyen programmal tudunk rá fájlokat másolni és törölni; állományokat szerkeszteni, közvetlenül mentést készíteni rá (és cserélgetni a lemezeket, ha szükséges). Egy probléma van csak, a *FreeBSD* nem tudja olvasni sem a Linux által készített *UDF2.0* fájlrendszert... de ez talán idővel megoldódik.

Ha kedvünk támad, akkor videó *DVD*-t is készíthetünk, ehhez egy tetszőleges videóállomány, és két program kell csak: az *Avidemux2* és a *DVDAuthor*.

Az *Avidemux2* egy videószerkesztő program, amely nevéből adódóan az *AVI* formátumra van felkészítve, bár olvas és ír nagyon sok más videóformátumot is.

A program részletes ismertetése meghaladja a cikk keretét, ezért csak a témára szorítkozok. Egy *DVD* lemezre *MPEG2* formátumban kell a videót elkészíteni, a hozzá tartozó hangokat pedig *AC3* formátumban. Több hangsávot is tudunk egyszerre kezelni (esküvői videó esetén az eredeti és a rákevert hangot például).

A legördülő menüknél kiválasztjuk a *DVD* és az *AC3* formátumot, a szűrőknél a videó méretét átállítjuk *DVD* méretre (720x576), majd bekapcsoljuk mind a két *Process* gombot. Ekkor a videót elmentve megindul a konvertálás *MPEG2*/*AC3* formátumra (amely egy két órás videó esetén egy 2GHz-es gépen is 10-11 óra lehet!).

A kész videót már csak rá kell varázsolni a *DVD* lemezre, erre a *DVDAuthor* programot tudjuk használni.

Ennek egy *XML* állomány kell, amelyben leírjuk, hogy mit szeretnénk készíteni:

```
<dvdauthor dest="/home/user/video/" jumppad="1">
    <vmgm />
    <titleset>
        <titles>
            <pgc>
             <vob file="/home/user/input.mpeg" />
            </pgc>
        </titles>
    </titleset>
</dvdauthor>
```

Ezek után egyszerűen csak el kell indítani:

```
$ dvdauthor -x dvdauth.xml
```

A megadott könyvtárban elkészül egy *DVD* adatstruktúra (`AUDIO_TS/VIDEO_TS`), amelyet tetszőleges íróprogrammal fel tudunk írni egy üres *DVD* lemezre. Ezzel el is készültünk, a kapott lemez bármilyen *DVD* lejátszóban működni fog.

Összegzés

Remélem senkit nem riasztottam el ezzel a pár oldalas rövid ismertetővel, amelyből kiviláglik, hogy a *FreeBSD* egyelőre nem (és előreláthatólag a későbbiekben sem) való munkaállomásnak. Ugyan szinte minden feladat megoldható, amelyeket egy átlagos felhasználó igényel, ennek ellenére sok munkánk adódhat azokkal a beállításokkal, amelyeket egy átlagosnál barátságosabb *Linux* terjesztés automatikusan elvégez (nyomtatósorok beállítása, hálózati hozzáférések beállítása, stb). Nem véletlen, hogy a *BSD* rendszerek inkább a kiszolgálók területén nagyobb arányú jelenlétet mutatnak, sok nagyobb cég *FreeBSD* operációs rendszert használ webkiszolgálásra és levelezésre; illetve tűzfalnak, útválasztónak és forgalomszabályzónak – a következő részekben megnézzük ennek okát.

Auth Gábor (auth.gabor@enaplo.hu)
Egy pécsi középiskolában informatikát és programozást oktat. Tíz éve botlott először a UNIX rendszerekbe, 7 év Linux használat után kapta el a FreeBSD lázat, amiből máig nem tudott kigyógyulni.

KAPCSOLÓDÓ CÍMEK

A FreeBSD projekt honlapja
➲ http://www.freebsd.org

A magyar FreeBSD honlap
➲ http://www.freebsd.hu

A magyar BSD honlap
➲ http://www.bsd.hu

A kézikönyv magyar fordítása
➲ http://www.enaplo.hu/FreeBSD/handbook/

Bevezetés a DHCP kiszolgáló használatába

TCP/IP alapú hálózatok központi beállítása és karbantartása – avagy ami unalmas, az a számítógép dolga.

Egyszer volt, hol nem volt, volt egyszer egy rendszergazda. Ennek a rendszergazdának egy kisebb helyi hálózatot kellett felügyelnie, főként olyan ügyfélgépekkel, amik azt a bizonyos másik operációs rendszert futtatták. Bár az a másik operációs rendszer felettébb megbízható, egyszer-egyszer mégis előfordult, hogy a türelmetlen felhasználók heves kattintgatásai és az internetről beáramló mérhetetlen mennyiségű kártevő miatt újra kellett telepíteni egy gépet.

Ilyenkor hősünk kezében egy telepítőlemezzel nekiesett a gépnek, és úgy másfél-két óra alatt varázsolt rá egy, a szűz hó érintetlenségének jeleivel büszkélkedő rendszert irodai csomaggal, és néhány fontos alkalmazással. A telepítés folyamata alapvetően nem kifejezetten szórakoztató, jópár szál cigaretta és nem kevesebb csésze kávé is elfogy egy ilyen művelet során. Viszont remek alkalom ez az alkalmazott módszer előnyeinek és hátrányainak mérlegeléséhez.

Történetünk rendszergazdája mérnök lévén, azt latolgatta, hogy vajon a telepítésnek melyik az a része, ami miatt feltétlenül az ő személyes beavatkozása szükséges. Magáról a rendszerről és az alkalmazásokról készíthető egy kép, ami egy írható ***DVD***-n pont elfér. Ennek felmásolását akár még a felhasználókra is rá lehetne bízni – na jó, azért ne essünk túlzásokba. Viszont ott van a hálózat beállításának kérdése, amit kénytelen-kelletlen, kézzel kell elvégezni.

Vagy mégsem? Milyen szép lenne a világ, ha az összes ***TCP/IP*** beállítás, címestül, átjáróstul, ***DNS***-kiszolgálóstul egy helyen, a sokat próbált ***Linux*** kiszolgálón volna, és az ügyfeleknek csak annyit kellene tudniuk, hogy ezeket az információkat le kell kérdezni. Hiszen ez már másnak is eszébe jutott! S ekkor hősünk fél óra alatt feltelepített egy ***DHCP*** kiszolgálót, és azóta élvezi a lustálkodás, valamint a ***bzflag*** édes örömeit.

A tanmese szereplői természetesen a képzelet szüleményei, ennek ellenére mindenkivel előfordulhat, hogy a munkahelyén azért rágják a fülét, mert a vállalati laptopnak látnia kellene az Internetet három telephelyen is, és senki sem akarja (vagy nem tudja) beállítani minden egyes alkalommal, hogy a helyi hálózaton most épp ki az átjáró. Erre a problémára jelent hosszú távú megoldást a ***DHCP***. Nézzük meg közelebbről, hogy mi is ez, és hogyan kell beállítani.

A ***DHCP*** a ***Dynamic Host Configuration Protocol***, azaz a dinamikus állomás beállító protokoll rövidítése. Az ***IETF (Internet Engineering Task Force)*** fejlesztette ki azzal a céllal, hogy egy nagyobb ***IP*** hálózat gépei egy központi helyről kérdezhessék le beállításaikat, ezáltal megkönnyítve a hálózat felügyeletét. A valamivel nagyobb múltra visszatekintő ***BOOTP***-vel visszafelé kompatíbilis, ám sok tekintetben tágabb lehetőségeket biztosít.

Az ***ISC (Internet Systems Consortium)*** által kiadott ***DHCP*** kiszolgáló, valamint ügyfél a jelenleg leginkább támogatott megvalósítás, ezért érdemes ennek a használatában elmélyedni. Kedvenc terjesztésünk, a ***Debian Linux*** csomagkezelőjében, a ***dselect***-ben kalandozva látható, hogy az ***ISC*** kiszolgálójának két változata is elérhető előre fordított csomag formájában. Az egyik a ***dhcp***, a másik a ***dhcp3-server*** nevet viseli. Előbbi a 2-es, míg utóbbi értelemszerűen a 3-as sorozatból származik.

A 3-as változatszámú ***DHCP*** kiszolgáló, illetve a hozzá tartozó ügyfél még tesztelés alatt áll, és a 2-es is igen szép szolgáltatás-választékkal áll rendelkezésünkre, ezért érdemes inkább ezt felrakni. Természetesen a forrásból történő telepítés sem nehéz, mindössze az ***ftp://ftp.isc.org/isc/dhcp/*** címről kell beszerezni a legújabb csomagot, kitömöríteni, majd `./configure && make && make install`. Ez tényleg ennyire egyszerű!

A ***Debian*** csomagleírása szerint a ***DHCP 2***-höz elég egy 2.0.32-es, a 3-ashoz már szükség van egy 2.2-es sorozatú Linux rendszermagra. Ez nem túl erős rendszerkövetelmény, amire azonban feltétlenül oda kell figyelni, az az, hogy a `CONFIG_PACKET` és a `CONFIG_FILTER` be legyen állítva. Ugyanakkor bizonyos 2.1-es sorozatú magokkal még így is gondok lehetnek, emiatt érdemes inkább legalább egy 2.4-es sorozatból származót használni.

Milyen gond származhat a régebbi kernelek használatából? A ***Windows*** korai változatainál, többek között ***Windows 95***-ös ügyfelek esetén, a helyes működéshez a ***DHCP*** kiszolgálónak képesnek kell lennie ***IP*** csomagot küldeni a ***255.255.255.255*** címre. Sajnálatos módon a ***Linux*** az ilyen célállomású IP csomagokat a helyi hálózat üzenetszórási címére irányítja. Így egy közönséges C típusú magánhálózatnál a ***255.255.255.255***-öt átírja ***192.168.0.255***-re.

Bizonyos esetekben ez a probléma megkerülhető. A legegyszerűbb, ha a rendszermag útvonalválasztó táblájába felveszünk egy sort, ami a ***255.255.255.255***-ös állomást a megfelelő hálózati csatolóhoz párosítja:

```
route add -host 255.255.255.255 dev eth0
```

Ez azonban nem mindig jelent gyógyírt a fenti gondra, mert egyes kernel változatok hibaüzenettel utasítják el a megadott *IP* állomásként történő bejegyzését. Még mindig becsaphatjuk a rendszermagot, ha a */etc/hosts* állományba felvesszük nevesítve a címet:

```
# /etc/hosts
255.255.255.255    mindenki
```

Ezután próbáljuk meg kiadni a következő parancsot:

```
route add -host mindenki dev eth0
```

Ha még ez sem válna be, próbálkozzunk az alábbival:

```
route add -net 255.255.255.0 dev eth0
```

Ezúton szeretnék elnézést kérni a – remélhetőleg – tőlem teljesen szokatlan ködös fogalmazásért. A fenti trükköket az *ISC DHCP* kiszolgálójának leírásában olvastam és sem ott, sem más dokumentációban nem találtam arra vonatkozó információt, hogy mikor melyik próbálkozás jelenti a megoldást. Ezért én azt javaslom, hogy mindenki, aki *DHCP* kiszolgáló telepítésére adja a fejét, és még nem frissített legalább 2.4-es változatú kernelre, előbb azt tegye meg.
Ha tűzfallal is rendelkezik leendő linuxos *DHCP* kiszolgálónk, néhány új szempontot figyelembe kell vennünk. Bármely *IP* címről, a 68-as *UDP* kapuról érkező üzenetszórt csomagokat át kell engedni, ha azok a 67-es *UDP* kapura érkeznek. Továbbá az ellentétes irányú kommunikációt hasonlóan engedélyezni kell. Egy tipikus *iptables/netfilter* beállítás lehet az alábbi:

```
iptables -I INPUT 1 -m state –state
➥ESTABLISHED,RELATED -j ACCEPT
iptables -I INPUT 2 -i eth1 -p udp –dport 67 -j
➥ACCEPT
iptables -I OUTPUT 1 -m state –state ESTABLISHED,
➥RELATED -j ACCEPT
```

A fenti példához szükség van az `ip_conntrack` modulra. Ez a három szabály azt eredményezi, hogy a tűzfal az eth1 csatolón, a 67-es *UDP* kapura érkező új kapcsolatokat fogadja el, valamint a kapcsolatkövetésnek köszönhetően az így kiépített kommunikációt engedélyezi. Itt a megszorítást a hálózati csatoló meghatározása jelentette. Fontos, hogy ilyenkor *IP* címre nem lehet szűrni, hiszen az ügyfelek pont azért kapcsolódnak a kiszolgálóhoz!
Nincs más hátra, mint a *DHCP* démon beállító-állományának szerkesztése. Alapértelmezésben */etc/dhcpd.conf* néven találjuk ezt a fájlt. Egy olyan szerver esetében, amelynek feladata mindössze egyetlen alhálózat kiszolgálása, nincs nehéz dolgunk. Tekintsük az alábbi beállítást, és mielőtt továbblépnénk, töprengjünk el azon, hogy mit eredményezhet.

```
# dhcpd.conf
#
option domain-name “lustakeavilag.hu”;
option domain-name-servers dns1, 192.168.0.10;

option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
option routers szerver;

default-lease-time 86400; # egy nap
max-lease-time 604800;    # egy het

subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.100 192.168.0.199;
}
```

A közönséges szöveges állományban szokás szerint a megjegyzések #-től sorvégéig tartanak. Minden beállítást tartalmazó sor végén pontosvessző áll. Előbb a globális beállítások következnek. Ilyen a `domain-name` paraméter, amely a tartománynevet határozza meg. Egy új ügyfél a *DHCP* kérésre kapott válaszban ezt a tartománynevet fogja látni, kivéve, ha ezt alább felül nem definiáljuk.
A következő `domain-name-servers` paraméterben, amely a *DNS* kiszolgálókat sorolja fel, azt a furcsaságot láthatjuk, hogy épp a névfeloldást végző számítógépre névvel hivatkoztunk. Ha a *DHCP* szerver képes feloldani ezt a nevet, akkor semmi gond, az ügyfélnek már a hozzá tartozó *IP* címet fogja közvetíteni. Ez mindössze a rendszergazdának egy kényelmi szempont, a teljesítményre semmilyen hatással nincs.
A `subnet-mask` az alhálózati maszkot, a `broadcast-address` az üzenetszórás címét adja meg. Az elsőre talán furcsának talált `routers` az átjárókat határozza meg. Természetesen itt is használhatunk neveket, ha azokat a *DHCP* szerver képes feloldani. Érdemes megfigyelni, hogy az eddigi beállításokat mindig egy `option` paraméter előzte meg. Ezek fontos, de a *DHCP* protokoll szerint elhagyható paraméterek. Az option nélkül állók azonban nem.
Ilyen a bérleti idők beállítására szolgáló `default-lease-time` és `max-lease-time`. A bérleti idő azt jelenti, hogy a *DHCP* által szolgáltatott információ bizonyos idő letelte után elévül. Ekkor az ügyfél felelőssége, hogy újra kérje ezeket az információkat. Ugyanakkor az ügyfél a *DHCP* kérésben meghatározhat egy kívánt bérleti időt, ameddig nem kell újabb kérést kiadnia. Ha nem fogalmaz meg ilyen kívánalmat, az alapértelmezett (default) bérletet kapja, ellenkező esetben a felső határig (max) a kiszolgáló megpróbálja kielégíteni a kérést.
Ez a két paraméter másodpercekben van kifejezve. A 86400 másodperc egy napot, a 604800 másodperc pedig egy hetet jelent. Ennyi ideig az ügyfél szabadon használhatja a kapott *IP* címet. Természetesen még a bérlet lejárta előtt bármikor lehetősége van azt megújítani. Ezt *Microsoft Windows 2000* és afölött parancssorban az alábbiakkal lehet elérni:

```
ipconfig /release
ipconfig /renew
```

A `release` elengedi a jelenlegi címet, a `renew` pedig megújítja. A `/all` kapcsolóval pedig a linuxos `ifconfig`-hoz hasonló kimenetet láthatunk a használatban lévő hálózati csatolókról. A példa *dhcpd.conf* végén egy alhálózat meghatározása lát-

ható. A hálózat neve, illetve az alhálózati maszk által azonosított alhálózatban a `range` paraméter segítségével lehet meghatározni azt a tartományt, amelyből az újonnan érkező ügyfeleknek *IP* címet lehet osztani. Egyszerűen egy alsó, illetve egy felső határt szab a felhasználható címeknek.
A fenti beállításokkal már elindulhat a kiszolgáló. Minden ügyfél tudni fogja, hogy melyik tartományba tartozik, kap egy egyedi *IP*-címet, és ha a többi paraméter is helyes, az Internetes szupersztráda által nyújtott korlátlan lehetőségekkel is élhet. Ha mindent jól csináltunk, *IP*-cím ütközéstől sem kell tartanunk a helyi hálózaton.
Ha figyelemmel kísérjük a rendszernaplót, láthatjuk, ahogy az ügyfelek egymás után fordulnak kéréssel a *DHCP* kiszolgálóhoz, és címet is kapnak szépen sorban. Azonban egy ilyen elrendezés esetén a dinamikus IP-cím kiosztás miatt problémás lehet behatárolni, hogy egy adott IP melyik géphez tartozik. Tegyük fel, hogy az egyik ügyfél egy hirtelen ötlettől vezérelve elkezdi *ICMP* üzenetekkel elárasztani a hálózatot. Ahhoz, hogy megtaláljuk, melyik gép volt az, a naplóhoz kell fordulnunk minden esetben.
További gondot jelenthet, hogy ezek után bárki kaphat *IP*-címet, és azonnal látja az egész hálózatot. Egy laptoppal rendelkező betolakodónak csak egy végpontot kell találnia, és máris osztozhat a hálózati erőforrásokon. A *DHCP* protokoll használatakor a hitelesítés egyetlen módja a hálózati kártyák fizikai címének figyelembe vétele. Ez az úgynevezett *MAC*-cím elvileg teljesen egyedi az egész világon és nem lehet két olyan hálózati eszköz, aminek megegyezik a *MAC*-címe.
Fűzzük hozzá meglévő ***dhcpd.conf***-unkhoz az alábbi sorokat:

```
group {
    use-host-decl-names on;
    host gizi {
        hardware ethernet 01:02:03:04:05:06;
        fixed-address 192.168.0.10;
    }
    host bela {
        hardware ethernet 0a:0b:0c:0d:0e:0f;
        fixed-address 192.168.0.20;
    }
}
```

A `group` csupán azt a célt szolgálja, hogy logikai egységbe foglalja a meghatározásokat. Így a paraméterek az összes, a csoportban szereplő elemre érvényesek lesznek. Például a `use-host-decl-names` egy olyan paraméter, amely előírja, hogy az azonosított számítógépek *DHCP* kérésére a számítógép nevét is el kell küldeni a válaszban. Így az ügyfeleknek a saját nevüket sem kell tudniuk, ezt az információt is megkapják a *DHCP* kiszolgálótól.
A `host` egy azonosítható állomást határoz meg. A `gizi` nevűnek a *MAC*-címe `01:02:03:04:05:06`, és a `192.168.0.10` *IP*-cím tartozik hozzá. Ez azt jelenti, hogy a `subnet` meghatározástól függetlenül, ha egy adott fizikai címmel rendelkező hálózati eszközről érkezik a kérés, a megadott IP-címet kell kiosztani. Ezáltal bár csak a szerveren tárolódik az összes hálózati beállítás, a két ügyfél állandó címmel rendelkezik, így könnyebb őket nyomon követni. Továbbá senki más nem bitorolhatja a megadott címeket.

Nagyon fontos, hogy a `subnet`-ben meghatározott tartomány ne érintse a `host`-okban foglaltakat. Ellenkező esetben előfordulhat, hogy egy ismeretlen *MAC*-című gép elfoglalja egy ismert *IP*-jét! Ezzel a módszerrel tehát elérhető, hogy minden ismert számítógép előre megadott, állandó IP címmel rendelkezzen. Adott esetben a `subnet` meghatározástól meg is válhatunk ezek után, de tudnunk kell, hogy ekkor a *DHCP* szerver visszautasít minden ismeretlen kérést.
Ezt figyelembe kell vennünk, ha hálózati kártyát cserélünk valamelyik ügyfélnél, vagy új számítógép érkezik a céghez. Nem mehetünk el szó nélkül a ***Microsoft Networks*** hálózat mellett sem. A ***Netbios*** nevek feloldásához érdemes a tartományvezérlőt ***WINS*** kiszolgálóvá tenni. Ez egy ***Samba*** kiszolgáló esetén egyszerűen az alábbi paraméter segítségével történik:

```
wins support = yes
```

Ezáltal a hálózat tallózása sokkal gyorsabb lesz, feltéve, hogy az ügyfelek tudják, kihez kell fordulni a ***Netbios*** nevekért. Ezt az információt is könnyen közzétehetjük új *DHCP* kiszolgálónkkal, az alábbi módon:

```
option netbios-name-servers 192.168.0.25;
```

A fenti sort a ***dhcpd.conf*** globális beállításai között kell elhelyezni, és egy újraindítás után az ügyfelek a következő *DHCP* kéréskor már a *WINS* kiszolgálóról is értesülni fognak.
Apropó, újraindítás. Jó volna, ha amikor ***dhcpd.conf***-hoz hozzáadunk egy új `host` bejegyzést, nem kellene újraindítani a démont. Erről hosszas viták folytak több levelezőlistán, és sajnos egyelőre nem várható változás ezzel kapcsolatban. Miután külső forrást sem lehet megjelölni a `host` meghatározásoknak, be kell érnünk azzal, hogy minden módosítás után újra kell indítani a `dhcpd`-t.
Ez a rövid leírás mindössze ízelítőül szolgált a *DHCP* lehetőségeinek bemutatására. Az interneten azonban számos jól használható leírás található, ha ez a cikk, illetve a kézikönyv lapok (`man dhcpd`, `man dhcpd.conf`) kevésnek bizonyulnának. Sok sikert a beállításokhoz és kellemes lustálkodást mindenkinek!

Fülöp Balázs (admin@guardware.com)
Imádja a Túró Rudit, a Debian Linuxot és a teheneket. Kedvenc írója Slawomir Mrożek. Leginkább a számítógépes hálózatok biztonsága érdekli. A BME VIK műszaki informatikus szak hallgatója.

KAPCSOLÓDÓ CÍMEK

- http://rfc.net/rfc2131.html
  A DHCP protokoll hivatalos leírása
- http://www.dhcp-handbook.com/dhcp_faq.html
  DHCP gyakran ismételt kérdések
- http://www.isc.org/index.pl?/sw/dhcp/
  Az ISC DHCP honlapja, számos remek linkkel

A PHP5 Reflection API

A sorozat előző részeiben megismerkedhettünk a PHP5 objektumközpontú újításaival, megtanulhattuk azok rendeltetésszerű használatát, illetve elmélyedhettünk a programnyelv által rendelkezésünkre bocsátott különleges tagfüggvények adta lehetőségekben. A cikksorozat záró akkordjaként az újdonságnak számító Reflection API-t szeretném bemutatni, amellyel lehetőségünk nyílik a már elkészített függvények, osztályok, metódusok, paraméterek, objektumok, stb. elemzésére futásidőben.

Mi is ez valójában?

Ez az ***API (alkalmazás programozási felület)*** nem más, mint egy osztálygyűjtemény, amelyet a ***PHP*** bocsát rendelkezésünkre – hasonlóan az ***Exception*** kivétel osztályhoz. Az egyes programelemek elemzéséhez a megfelelő elemző osztályt kell kiválasztanunk, amelyek mindegyike a ***Reflection*** ősosztályból öröklődnek. A szolgáltatás igénybe vételéhez nincs is más dolgunk, mint a megfelelő konstruktor-paraméterekkel példányosítani az osztályokat, majd az adott példány metódusain keresztül elérni a paraméterek szerinti programelemek összes létező tulajdonságát. A továbbiakban megnézzük, hogy milyen osztályokkal van dolgunk, ezek mire használhatók – néhány rövid példával illusztrálva őket.

Sorban az első: ReflectionFunction

A nevéből már kitalálhattuk, hogy itt bizony az egyes függvények nyilvános és rejtett tulajdonságait nézhetjük meg közelebbről. Az itt leírt használati módszer egyébként az összes többi osztály esetében is hasonló, sőt, alapjaiban ugyanez, csak a későbbi osztályokban más és más (általában egyre nagyobb) az elérhető elemző tagfüggvények köre. Néhány fontosabb metódust az *1. táblázat* tartalmaz.

Mivel ez így még mindig homályosan hangzik kissé, lássunk egy példát az általános használatra, amelyben minden fenti metódusra szerepel egy-egy példa. Bár ez konkrétan a ***ReflectionFunction*** osztály hétköznapi viselkedését szemlélteti, a többi osztálynál sem mutatkozik jelentős eltérés, így hát valamiféle általános igazságként tekintsünk erre a programra.

1. táblázat *A ReflectionFunction néhány fontosabb metódusa*

`getName()`	Visszaadja a függvény nevét.
`getFileName()`	Megmondja, hogy mi a neve annak a fájlnak, amelyben szerepel.
`getStartLine()`	Melyik sortól kezdődik a függvény implementációja.
`getEndLine()`	Melyik sorban végződik a függvény implementációja.
`isInternal()`	Igaz, ha belső, *PHP*-s függvényről van szó.
`isUserDefined()`	Igaz, ha felhasználói függvényről van szó.
`getDocComment()`	Visszaadja a függvényhez tartozó *PHPDoc/JavaDoc*-nak megfelelő dokumentáció jellegű megjegyzést.
`invoke()`	Meghívja a függvényt a paraméterben szereplő paraméterekkel, majd visszatér a függvény visszatérési értékével.
`export()`	Statikus függvény, amely megjeleníti az alapértelmezett kimeneten az elemzett függvény szerkezetét, felépítését, általános adatait

```
<?php
//----A deklarációs rész-------------------------
/**
 * Szabványos módon kommentezett függvény,
 * amely négyzetre emeli a paramétert.
 *
 * Ez a megjegyzés phpdoc/javadoc alapú
 * (megfelelő), s a Reflection API-n
 * keresztül is lekérdezhető
 * @param      int     Bementő paraméter
 * @return     int     Visszatérési érték
 */
function negyzet($i=0) {
    return $i*$i;
}
```

```
//----Az elemző/kiíratórész----------------------
// létrehozzuk az elemző objektumot, amelyen
// keresztül a kívánt paraméterek
// lekérdezhetők – EZ ITT A LENYEG
$func = new ReflectionFunction('negyzet');

echo "<pre>";

//alap információ kiíratása
echo
"==========================================\n";
echo "Általános információk\n";
echo
"==========================================\n";
printf(
    "A függvény neve: %s\n".
    "A függvény típusa: %s\n".
    "A függvény deklarálásának helye:\n".
    "   fájl: %s\n".
    "   deklaráció kezdete: %d. sor\n".
    "   deklaráció vége: %d. sor\n",
   $func->getName(),
   $func->isInternal() ? 'belső' : 'fehasználói',
   $func->getFileName(),
   $func->getStartLine(),
   $func->getEndline()
);

// A dokumentáció jellegű komment kiíratása
echo
"\n==========================================\n";
echo "PHPDoc/JavaDoc stílusú komment:\n";
echo
"==========================================\n";
printf("%s\n", var_export($func->getDocComment(),
➥1));

//A függvény hívása
echo
"\n==========================================\n";
echo "Függvényhívás eredménye:\n";
echo
"==========================================\n";
var_dump($func->invoke(1));
var_dump($func->invoke(2));
var_dump($func->invoke(3));

//A paraméterek
echo
"\n==========================================\n";
echo "A függvény paraméterei:\n";
echo
"==========================================\n";
var_dump($func->getParameters(3));

//Az export metódus eredménye
echo
"\n==========================================\n";
echo "A függvény szerkezete:\n";
echo
```

2. táblázat *A ReflectionFunction néhány fontosabb metódusa*

`getClass()`	Visszaadja, hogy mely osztály példányát várjuk paraméterként (`NULL`, ha alaptípusról van szó) *ReflectionClass* típus formájában, amely azonnal további elemzést tesz lehetővé.
`getDefaultValue()`	Visszaadja a paraméter alapértelmezett értékét.
`allowsNull()`	Igaz, ha null érték megengedett a paraméter használata során.
`isPassedByReference()`	Igaz, ha referencia szerinti paraméterátadás formájában használjuk a paramétert (ha objektumot adunk át, ez mindig igaz lesz).
`isOptional()`	Igaz, ha opcionális paraméterről van szó.

3. táblázat *A ReflectionFunction osztálytól különböző néhány fontosabb metódus*

`getConstructor()`	Visszaadja az osztály konstruktorát a *ReflectionMethod* osztály példányaként
`getMethotds()`	Visszaadja az osztály metódusait a *ReflectionMethod* osztály példányaiként egy tömbben – a konstruktort is beleértve
`getProperties()`	Visszaadja az osztálytulajdonságokat a *ReflectionProperty* osztály példányaiként egy tömbben
`getConstants()`	Visszaadja az osztály konstansait egy tömb értékeiként
`isInterface()`	Igaz, ha felületet elemzünk
`isAbstract()`	Igaz, ha az osztály elvont
`isFinal()`	Igaz, ha nem örökíthető az osztály tovább
`isInstantiable()`	Igaz, ha példányosítható az adott osztály (pl. elvont osztály nem példányosítható)
`getParentClass()`	Visszaadja a szülő osztályt a *ReflectionClass* osztály példányaként
`isSubClassOf()`	Igaz, ha a paraméterben szereplő osztály alosztája a vizsgált osztály
`isIterateable()`	Igaz, ha az osztály iterálható (*PHP5*-ben ugyanis az osztályokon, objektumokon végigmehetünk egy **foreach**-el, mintha mondjuk tömbök lennének, ahol a tömb elemei az egyes tulajdonságok, tagfüggvények, konstansok, stb.)

```
"==========================================\n";
echo ReflectionFunction::export('negyzet');

echo "</pre>";
?>
```

A program futási eredményét terjedelmi okokból nincs értelme közölni, de mint az magából a kódból látható, egyfajta összefoglaló elemzést készít.

Sorban a következő: ReflectionParameter

Szorosan kapcsolódik az előző osztályhoz ez a függvény és tagfüggvény paramétereket elemezni tudó osztály. A kapcsolat valójában abban áll, hogy a paramétereket, mint osztályokat a `ReflectionFunction::getParameters()` metódusával kaphatjuk meg, nem példányosítással, mint az előző esetben. A fenti függvény tehát ***ReflectionParameter*** típusú elemek tömbjével tér vissza, amelyek sorban a függvény vagy tagfüggvény egyes paramétereit képviselik. A teljesség igénye a *2. táblázatban* felsoroltam néhány fontosabb tagfüggvényt.
A fenti példánál maradva elemezzük a függvény egyetlen paraméterét a szabványos módon:

```
//tovabbi elemzes
echo
"\n==========================================\n";
echo "A függvény paramétereinek elemzése:\n";
echo
"==========================================\n";
foreach ($func->getParameters() as $i => $param) {
    printf(
        "Paraméter: #%d\n".
        "----------------------\n".
        "       Neve: %s\n".
        "       Osztálya: %s\n".
        "       Alapértelmezett érték: %s\n".
        "       Null érték megengedett?: %s\n".
        "       Referenciaként átadva?: %s\n".
        "       Opcionális?: %s\n".
        "\n",
        $i,
        $param->getName(),
        var_export($param->getClass(), 1),
        var_dump($param->getDefaultValue()),
        var_export($param->allowsNull(), 1),
        var_export($param->isPassedByReference(),
        ➥1),
        var_export($param->isOptional(), 1)
   );
}
```

Jól látható, hogy a paraméter elemzés nem áll többől, mint végig menni a visszakapott paramétereken, s mindegyiknél valami hasonlót kell eljátszani, mint a függvény elemzésénél.

Osztályok vizsgálata a ReflectionClass segítségével

Felépítése többnyire a metódusok számában különbözik a ***ReflectionFunction*** osztályétól, s a használat során is leginkább a gazdagabb funkciók körében kell keresnünk a különbségeket. Igazából nem is csak osztályok, de ***felületek (interface)*** vizsgálatára is alkalmas. Az analízis tényleg teljes körű: Megmondja, hogy melyik osztályból öröklődik, melyik felületet valósítja meg, elvont-e (abstract), milyen metódusai vannak - ha vannak, példányosítható-e az osztály, visszakaphatjuk a konstruktort, destruktort (természetesen a ***ReflectionMethod*** osztály példányaként a további elemzés céljából), az osztálytulajdonságokat (osztályváltozókat) hasonló elemezhető formában, és még sorolhatnám. Néhány, a ***ReflectionFunction*** osztálytól különböző lényeges tagfüggvény leírását a *3. táblázat* tartalmazza.
Példaként álljon itt egy a ***ReflectionFunction*** osztálynál szereplő példához hasonló osztály-elemző program:

```
<?php
//----A deklarációs rész--------------------------
class Negyzet {
    protected $oldal = 0;

    public function __construct($oldal) {
        $this->oldalHossztBeallit($oldal);
    }

    public function oldalHossztBeallit($ertek) {
        $this->oldal=$ertek;
    }
}

class Rombusz extends Negyzet {

    protected $szog = 0;

    public function __construct($oldal,$szog) {
        parent::__construct($oldal);
        $this->szogetBeallit($szog);
    }

    public function szogetBeallit($szog) {
        $this->szog=$szog;
    }

    public function terulete() {
        echo $this->oldal*sin($this->szog)
        ➥*$this->oldal;
    }
}
$class = new ReflectionClass('Rombusz');

echo "<pre>";

//alap információ kiíratása
echo
"==========================================\n";
echo "Általános információk\n";
echo
"==========================================\n";
printf(
    "Az osztály neve: %s\n".
    "A osztály jellemzői: %s, %s, %s, %s\n".
    "A függvény deklarálásának helye:\n".
```

```
    “   fájl: %s\n”.
    “   deklaráció kezdete: %d. sor\n”.
    “   deklaráció vége: %d. sor\n”,
    $class->getName(),
    $class->isInternal() ? ‘belső’ :
    ➥‘felhasználói’,
    $class->isAbstract() ? ‘elvont’ : ‘nem elvont’,
    $class->isFinal() ? ‘ final’ : ‘gyermekből
    ➥felülírható’,
    $class->isInterface() ? ‘felület’ : ‘osztály’,
    $class->getFileName(),
    $class->getStartLine(),
    $class->getEndline()
);

// Szülő osztály dolgai
echo
“\n=========================================\n”;
echo “Szülő osztály szerkezete\n”;
echo
“=========================================\n”;
Reflection::export($class->getParentClass());

// Megvalósított felületek
echo
“\n=========================================\n”;
echo “Megvalósított felületek\n”;
echo
“=========================================\n”;
printf(“%s\n”, var_export($class->getInterfaces(),
➥1));

// Osztálytulajdonságok
echo
“\n=========================================\n”;
echo “Osztálytulajdonságok\n”;
echo
“=========================================\n”;
printf(“%s\n”, var_export($class->getProperties(),
➥1));

// Metódusok
echo
“\n=========================================\n”;
echo “Tagfüggvények\n”;
echo
“=========================================\n”;
printf(“%s\n”, var_export($class->getMethods(),
➥1));

echo “</pre>”;
?>
```

Szeretném felhívni a figyelmet a következő sorra: `Reflection::export($class->getParentClass());`. A *Reflection* ősosztály rendelkezik egy olyan, mindegyik gyermekéből elérhető statikus metódussal, amely előre meghatározott formában a szabványos kimenetre küldi a paraméterként átadott osztály, függvény, paraméter, stb. szerkezetét. A fenti programban ezt arra használtuk, hogy megmutassuk az ősosztály felépítését.

Ha tovább szeretnénk menni, némi rekurzió alkalmazásával elérhetjük, hogy nem csak a közvetlen őst, de az osztály összes felmenőjét megvizsgáljuk, amelyet alaposan megkönnyít, hogy az ősosztály is egy ilyen *ReflectionClass* osztály példányaként érkezik, aminek szintén van `getParentClass()` metódusa, tehát nekünk szinte már semmi dolgunk.

4. táblázat *A ReflectionMethod néhány fontosabb tagfüggvénye*

`isFinal()`	Igaz, ha a tagfüggvény a gyermekekben nem írható felül.
`isAbstract()`	Igaz, ha a tagfüggvény elvont.
`isPrivate()`	Igaz, ha a tagfüggvény csak az osztályon belül érhető el (hasonlóan: `isProtected()`, `isPublic()`).
`isStatic()`	Igaz, ha a metódusunk statikus.
`isConstructor()`	Igaz, ha az aktuális metódus az osztály konstruktora (hasonlóan: `isDestructor()`).

A ReflectionMethod osztály

Ez az előző bekezdésben már emlegetett osztály az egyes tagfüggvények elemzésére szolgál. Keletkezését tekintve tudni kell róla, hogy a *ReflectionFunction* osztályból öröklődik, így mind működésében, mind feladatában igen hasonlít az ősére, azzal az apró különbséggel, hogy tartalmaz néhány tagfüggvényt a metódusok függvényekhez képest plusz tulajdonságainak lekérdezésére. A fontosabb tagfüggvények leírását a *4. táblázat* tartalmazza.

Egészítsük ki az előző példánkat az alábbi tagfüggvényelemző kódrészlettel:

```
//Metódusinformáció
echo
“\n=========================================\n”;
echo “Metódusok adatai\n”;
echo
“=========================================\n”;
foreach ($class->getMethods() as $i => $method) {
    printf(
        “Metódus: %d\n”.
        “----------------------\n”.
        “       Neve: %s\n”.
        “       Tulajdonságai: %s, %s, %s, %s\n”.
        “       Típusa: %s\n”.
        “       Deklarálásának helye:\n”.
        “               fájl: %s\n”.
        “               deklaráció kezdete: %d.
                        ➥sor\n”.
        “               deklaráció vége: %d.
                        ➥sor\n”.
      “\n”,
```

5. táblázat *A ReflectionProperty néhány fontosabb metódusa*

`isPrivate()`	Igaz, ha az osztálytulajdonság csak az osztályon belül érhető el (hasonlóan: `isProtected()`, `isPublic()`).
`isStatic()`	Igaz, ha az az osztálytulajdonság statikus.
`isDefault()`	Igaz, ha már a fordítás során deklarált a paraméter, hamis, ha csak futásidőben tesszük ezt meg.
`getValue()`	Lekérdezhetjük az értékét.
`setValue()`	Beállíthatjuk a tulajdonság értékét, amely akkor lehet hasznos, ha a `ReflectionMethod::invoke()` metódusával szeretnénk meghívni a függvényt, és az esetleg használja valamelyik osztálytulajdonság értékét is.

```
            $i,
            $method->getName(),
            $method->isAbstract() ? 'elvont' :
            ➥'nem elvont',
            $method->isFinal() ? 'final' : 'gyermekből
            ➥felülírható',
            $method->isStatic() ? 'statikus' : 'nem
            ➥statikus',
            $method->isPublic() ? 'public' : $method->
            ➥isPrivate() ? 'private' : 'protected',
            $method->isInternal() ? 'belső' :
            ➥'fehasználói',
            $method->getFileName(),
            $method->getStartLine(),
            $method->getEndline()
        );
}
```

Eredményül megkapjuk az osztály jellemzői alatt a hozzá tartozó metódusok (örökölt és nem örökölt egyaránt) tulajdonságait.
Ennél természetesen még tovább is mehetnénk.
A *ReflectionParameter* osztály esetében már láttunk arra példát, hogy a *ReflectionFunction* osztály által visszaadott paramétereket hogyan kell 'bejárni', ezáltal elemezni. Ezt itt is megtehetjük, ha másért nem, hát gyakorlásképpen, és hozzáírhatjuk az egyes metódusokhoz, hogy milyen tulajdonságú paraméterekkel rendelkeznek. Kezd körvonalazódni, hogy ezek az osztályok egymással egész kis láncolatot alkotnak, jelentősen egyszerűsítve nekünk, fejlesztőknek a munkáját.

A ReflectionProperty osztály

Mint tudjuk, egy osztálynak nem csak tagfüggvényei, hanem tulajdonságai is vannak. Az osztály-elemző példában ezt elintéztük annyival, hogy rázúdítottuk a kimenetre a `ReflectionClass::getProperties()` metódusának

visszatérési értékét. A kimenetből láthattuk is, hogy ezek ***ReflectionProperty*** típusúak, így a már megszokott módon végigmehetnénk a paramétereken, hogy kicsit azokat is elemezzük. Mielőtt ezt mentenénk, nézzük, milyen tulajdonságokat kérdezhetünk le egy ilyen osztálytól (*5. táblázat*). Most pedig cseréljük ki az osztálytulajdonságok elemzését végző részt az alábbi kódrészletre:

```
// Osztálytulajdonságok
echo
"\n=========================================\n";
echo "Osztálytulajdonságok\n";
echo
"=========================================\n";
foreach ($class->getProperties() as $i => $method)
{
    try {
        $value=var_export($method->getValue(),1);
    } catch (Exception $ex) {
        $value='private vagy protected,
        ➥nem lekérdezhető';
    }

    printf(
        "Osztálytulajdonság: %d\n".
        "------------------------\n".
        "       Neve: %s\n".
        "       Tulajdonságai: %s, %s, %s\n".
        "       Értéke: %s".
       "\n",
        $i,
        $method->getName(),
        $method->isStatic() ? 'statikus' :
        ➥'nem statikus',
        $method->isPublic() ? 'public' : $method->
        ➥isPrivate() ? 'private' : 'protected',
        $method->isDefault() ? 'fordítási időben
        ➥deklarálva' : 'futásidőben megadva',
        $value
   );
}
```

Egyetlen említésre méltó elem a kódrészletben a kivételkezelési rész. Ha ugyanis nem publikus értékkel dolgozunk, akkor egy kivétellel jelzi az osztály, hogy ez bizony nem fog menni. Általában is igaz, hogy ha olyan műveletet szeretnénk végezni, amely valamilyen ok miatt nem lehetséges, s általános esetben programhibát okozna, arról kivételt generál a ***PHP***, amelyet nekünk kell lekezelnünk.

Mondhatjuk, hogy a végére értünk a legfőbb osztályoknak (természetesen akad még ilyen funkciójú ***Reflection*** osztály), s most nem árt feltennünk a kérdést, hogy mégis mire jó ez, mégis miért dolgoztunk ennyit?

A válasz tömören az, hogy minden esetben hasznos lehet, ahol általános, vagy ismeretlen eredetű osztályok, függvények együttműködését kell garantálnunk, például ha több fejlesztő dolgozik együtt, vagy ha olyan átfogó „anyaosztályokat” készítünk, amelyekkel a programozói akadályok sokféleségét kezelni tudjuk.

A konkrét példáknál maradva előfordulhat, hogy olyan osztályokkal dolgozunk, amelyek többfélék lehetnek – az internetes környezetben úgyis gyakran fordulnak elő ún. félstrukturált adatok, amelyek vagy ilyenek, vagy olyanok, az mindig az adott körülményektől függ. Meglehetősen nagy annak a valószínűsége, hogy ugyanazt az általunk várt bemenetet többféle osztály is képviselheti, s nekünk a többféleség függvényében más és más metódusokat kell meghívnunk. Ekkor lehet hasznos vagy az osztály elemzése, vagy a lehetséges metódusok hívogatása, kezelése kivétellel, ezáltal annak kitapogatása, hogy vajon milyen bemenettel is van dolgunk. Enélkül elképzelhetetlen, hogy ne kapjunk programhiba-üzenetet, amit ugye senki sem szeret, ráadásul a programunkra sem fogható rá, hogy tökéletesen működik.

Ennél különlegesebb esett, ha szeretnénk egy automatikus dokumentáció-készítő programot csinálni, amely fejlesztői leírást készít minden általunk készített programelemre, méghozzá egyformán és gyorsan. Minden fejlesztő utál dokumentációt készíteni, pedig ha van, az a későbbiekben megoldást jelenthet egy rakás problémára. Normális esetben ez úgy lehetséges, ha elemezzük a ***php*** fájlokat, tehát írnunk kell egy kisebb fajta ***PHP*** motort – mondanom sem kell, mekkora emberi ráfordítással. Ha azonban kihasználjuk a ***Reflection API*** nyújtotta lehetőségeket, akkor már az itt szereplő kódrészlet nem túl jelentős átalakításával is kaphatunk például ***HTML*** alapú dokumentáció-készítő programocskát, amivel ugyan kézzel és egyenként kell végigmennünk az osztályokon, objektumokon, de legalább a dokumentációt nem nekünk kell megcsinálni.

Egyszerűsíthetjük is a dolgot, s annak a bizonyos `getDocComment()` metódusnak a visszatérési értékét elemezve készíthetünk egy egyszerű kis fejlesztői leírást, amelyet odaadhatunk a fejlesztői csapat többi tagjának, ezáltal csupán a kommentezéssel megússzuk a dokumentáció készítését, ha ügyesek és körültekintőek vagyunk. Megjegyezném, hogy ilyen programok már léteznek, a fenti elvárásokat teljesíti például a ***PHPDocumentor (http://www.phpdoc.org/)***, amely egyike a legjobbaknak

Láthatjuk, hogy nem épp hétköznapi igények megoldására használhatjuk igazán hatékonyan a ***Reflection API***-t, de azért számtalan olyan helyzetet találunk, amelynél szintén nagy könnyebbséget jelent a használata. Sajnos nem állunk túl jól a témáról fellelhető leírást illetően.

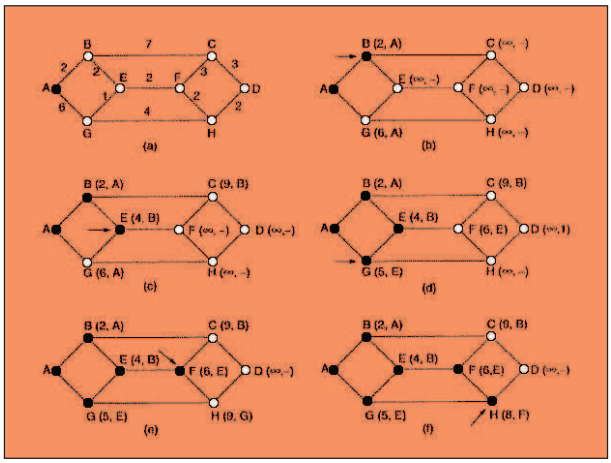
A ***Google*** által első helyeken felkínált találatok mind-mind a ***http://www.php.net*** weboldalon található dokumentációt követik, ami viszont nem valami bőbeszédű, sok helyen pedig messze nem teljes körű. Öröm az ürömben, hogy a ***Reflection API*** kiválóan alkalmas ám önmaga elemzésére is, ezáltal a hiányzó információkhoz viszonylag könnyedén hozzájuthatunk egy-két ***PHP*** parancs kiadásával – jelen cikk írása során is előfordult párszor, hogy a tárgyalt téma képviselőjére bíztam önmaga feltérképezését.

Komáromi Zoltán
(komi@kiskapu.hu)
23 éves, a BME hallgatója, mellette PHP-programozóként dolgozik. Kedvenc területe a multimédia.

Számítógép hálózatok (15. rész)

Statikus forgalomirányítás, távolságvektor alapú forgalomirányítás, megosztott láthatár.

Tovább ismerkedünk a forgalomirányító algoritmusokkal, először megnézünk pár egyszerűbb, statikus eljárást, majd olyanokat, amelyek az alhálózati topológián kívül más tényezőket is figyelembe vesznek.
Az alhálózatnak törekednie kell arra, hogy a csomagokat a lehető leggyorsabban juttassa el rendeltetési helyére. Erre a legkézenfekvőbb megoldás az, ha a csomag számára a forrás és a cél között fellelhető legrövidebb utat jelöli ki. Ez az elnevezés azonban félrevezető, mivel az alhálózatbeli legrövidebb út nem biztos, hogy egybeesik a „földrajzi” értelemben vett legrövidebb úttal.
A hálózat két pontja közötti távolságot ugyanis nem csak azzal jellemezhetjük, hogy az adatoknak milyen hosszú kábelen (vagy egyéb más csatornán) kell áthaladniuk. A távolságot definiálhatjuk például az ugrások számával, azaz hogy a csomag hány útválasztó érintésével jut el a célállomásáig. Ugyanakkor az is elképzelhető, hogy a csomagokat a legkisebb kommunikációs költségű vonalakon keresztül szeretnénk áramoltatni. Ilyenkor a legrövidebb útnak a legolcsóbb útvonalat tekintjük. Sőt, az általános igény az, hogy ezeket a paramétereket kombinálni lehessen, azaz a legrövidebb út a távolság és a kommunikációs költség mellett az általános terheltség, a sávszélesség és az átlagos sorbanállási idő függvényeként legyen kiszámolva.
Ahhoz, hogy ezt az utat számítógépek segítségével (azaz matematikai úton) meghatározhassuk, a feladatot általánosítanunk kell úgy, hogy azt a nem túlzottan okos gépek is megértsék. Az előző részben az alhálózatot egy gráffal illusztráltuk, amelynek csomópontjai az útválasztók voltak. Két csomópontot akkor kötöttünk össze éllel, amikor a pontok által illusztrált útválasztók között létezett fizikai összeköttetés.
Most ezt a modellt egészítjük ki úgy, hogy minden élhez hozzárendelünk egy értéket, amelyet az adott élhez tartozó súlynak nevezünk. Ez az érték lesz az él által összekötött két csomópont távolsága. (Ezt a gyakorlatban a sávszélesség, a késleltetési idő, az átlagos sorbanállási idő és egyéb paraméterek függvénye szerint számolják ki. A súlyok meghatározásának módja azonban nem érinti a megoldás menetét).
Az eredeti feladatot tehát úgy fogalmazhatjuk át, hogy azt az A és B közötti utat keressük, amelyben a benne szereplő élek súlyainak az összege a lehető legkisebb. Ennek a feladat megoldására sok algoritmus létezik, mi most megnézzük ezek közül az egyik legnépszerűbbet.



1. ábra A Dijkstra algoritmus működése

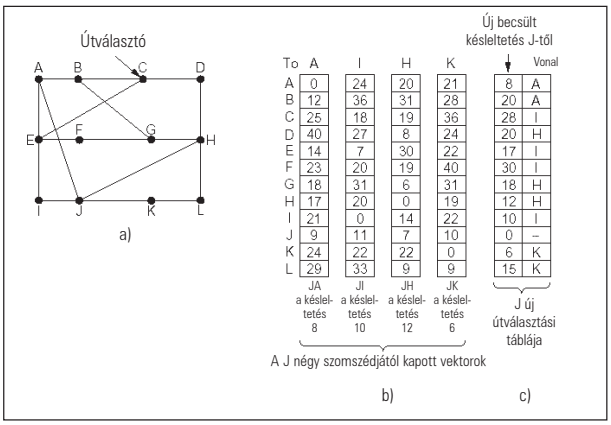
A legrövidebb útvonal (shortest path) algoritmus

Ez az algoritmus ***Dijkstra***, holland származású matematikus nevéhez fűződik (akit algoritmusai mellett az operációs rendszerek világában felhasznált ötletei (például a szemaforok) is ismerté tettek). Az algoritmus egy adott forrástól elkezdve folyamatosan keresi a célhoz vezető legrövidebb utakat. Minden csomóponthoz egy címkét rendel, amelyben eltárolja, hogy a kérdéses csúcs az eddig megtalált legrövidebb úton haladva milyen messze található a forrástól. Először persze egy utat sem ismerünk, így a címkék értéke végtelen. Ahogy telik az idő, az algoritmus egyre jobb utakat talál, így a címkék értéke folyamatosan változhat.
Amikor kiderül, hogy egy csomóponthoz nem lehet a már megismertnél rövidebb úton eljutni, akkor annak a címkéje véglegessé válik, értéke a továbbiakban nem módosul.
Az algoritmus működését az 1. ábrán szemléltetjük. Az „a”-részben látjuk az alhálózatunk felépítését, illetve az élek súlyait, azaz a csomópontok közötti távolságokat. A feladat az A-ból D-be vezető legrövidebb út megtalálása, ennek menetét láthatjuk az ábra többi részén. A már végleges címkével rendelkező csúcsokat besötétített karikával jelezzük.
Kezdetben csak az A csomópontot ismerjük, így azt nyugodtan besötétíthetjük. Ezután megnézzük, hogy az A csúcs mely más csúcsokkal van összekötve, jelen esetben a B és a G csomópontokkal. E két csúcs címkéjét átírjuk az

A-tól vett távolságukkal, azaz B címkéjének új értéke 2, G-nek pedig 6 lesz. Ahhoz, hogy a végeredményül kapott utat rekonstruálhassuk, szükséges, hogy a címkékben a távolságon kívül feltüntessük azt is, hogy az iménti vizsgálatot melyik csúcsból végeztük. Ezt a csúcsot az adott lépés munkacsomópontjának nevezzük (az ábrán egy nyíllal jelöljük). Ez az első lépésnél mindig a forrás, tehát az A csúcs. Ezután az ideiglenes (még be nem satírozott) címkék közül kiválasztjuk azt, amelynek az értéke a legkisebb. Ez most a B csúcs. A B címkéjét állandóra állítjuk, és megkeressük a szomszédait. Ha B egy szomszédjának címkéje kisebb, mint a B címkéje, és a vizsgálandó csúcs súlyának összege, akkor új rövidebb utat találtunk, és a csomópontot újra címezzük. Ha megvizsgáltuk B összes szomszédját, akkor megint kiválasztjuk az ideiglenes címkék közül a legkisebbet, és az eljárást ugyanígy folytatjuk.
Az algoritmus egészen addig nem áll le, amíg az összes utat meg nem találta. Fontos megjegyezni, hogy ez csak olyan gráfok esetén alkalmazható, amelyek nem tartalmaznak negatív súlyú éleket.

Elárasztás (flooding)

Egy másik egyszerű algoritmus az elárasztás. A feladat csupán annyi, hogy a bejövő csomagot az útválasztó minden szomszédjának továbbküldje (kivéve persze annak, akitől a csomag érkezett). Az egyetlen megoldandó probléma a kettőzött csomagok kezelése. Könnyen belátható, hogy beavatkozás nélkül minden csomagból végtelen számú példány keringene az alhálózaton.
Két módszer is ismert, amely segítségével megmenthetjük alhálózatunkat attól, hogy belefulladjon a csomagáradatba. Az egyik elképzelés szerint minden csomag fejlécének kéne tartalmaznia egy úgynevezett ugrásszámlálót, amelynek értéke mindig eggyel csökken, ahányszor a csomag áthalad egy útválasztón. Amikor a számláló eléri a nullát, a csomag megsemmisül. Az egyetlen kérdés az, hogy mekkora legyen a számláló kezdeti értéke. A módszer akkor a leghatékonyabb, ha az ugrásszámláló a forrás és a cél közötti távolság értékéről indul. Ez azonban nem mindig ismert, így a számlálót úgy kell beállítanunk, hogy a csomag a legrosszabb esetben is elérje célját. Ilyen érték lehet például az alhálózat teljes átmérője.
A másik megoldás az, ha minden útválasztó észben tartja, hogy mely csomagokkal végzett már elárasztást. Annak érdekében, hogy a csomagok azonosíthatóak legyenek, az útválasztó a beérkező csomagba – mielőtt tovább küldené a szomszédságának – egy azonosítószámot helyez. Minden útválasztó az összes portjához egy listát rendel, amelyben feljegyzi, hogy az adott forrástól eddig milyen sorszámú csomagokat kapott. Ha egy beérkező csomag még semelyik listában sem szerepel, akkor az útválasztó elvégzi az elárasztást, egyébként pedig eldobja azt. A probléma csak az, hogy a listák mérete folyamatosan növekszenek, így azok előbb vagy utóbb nem fognak elférni a memóriában. Az útválasztók ezért szomszédaikhoz lista helyett inkább egy számlálót rendelnek. A forrás a sorszámokat egyesével, növekvő sorrendben osztja ki, így ha a számláló értékét mindig az utolsó elárasztásra került csomag sorszámára állítva a beérkező csomagokról el tudjuk dönteni, hogy másodpéldányok-e vagy sem (ha a sorszám kisebb mint a számláló aktuális értéke, akkor biztosan másodpéldány).



2. ábra Távolságvektor-alapú forgalomirányítás
a) egyetlen alhálózat b) J útválasztóhoz a szomszédoktól érkező késleltetési vektorok c) J új forgalomirányító táblázata

Az elárasztás módszere kétségtelenül nem egy hatékony eljárás. Valamit javíthatunk a dolgon azzal, ha az elárasztást szelektíven végezzük. Ez alatt azt értjük, hogy a csomagokat csak azokra a vonalakra továbbítjuk, amelyek hozzávetőlegesen a cél felé tartanak. Általában felesleges egy keleti irányban lévő cél felé tartó csomagot a nyugatra mutató portokon is elárasztani.
Nehéz elképzelni, hogy lehet haszna egy olyan forgalomirányító algoritmusnak, amely az alhálózaton átmenő csomagok másolatainak tömkelegét állítja elő, és szerteküldi azokat a hálózat minden szegletébe. Ennek az eljárásnak azonban van pár egyedülálló képessége. Ilyen például az, hogy akár atomtámadás ellen is véd. Ha ugyanis hirtelen útválasztók tucatjai válnak működésképtelenné, a csomagok, igaz más útvonalon, de így is célba érkezhetnek. Egy katonai hálózat esetében ez egy hasznos tulajdonságnak bizonyulhat.
Az elárasztás igazi jelentősége azonban nem a gyakorlati felhasználásában rejlik, hanem abban, hogy összehasonlítási alapul szolgálhat más algoritmusok számára. Ez az eljárás ugyanis a csomagok számára az összes lehetséges utat egyszerre választja ki, amelyek között megtalálható a legrövidebb út is. Ez azt jelenti, hogy nem létezik olyan más algoritmus, amely ennél rövidebb késleltetési időt eredményezne (persze leszámítva az elárasztási műveletek által keltett késleltetést).

Távolságvektor alapú forgalomirányítás

Eddig csak statikus algoritmusokkal foglalkoztunk, amelyek döntéseikben kizárólag az alhálózat topológiája játszott szerepet. A továbbiakban olyan eljárásokat veszünk szemügyre, amelyek figyelembe veszik a hálózat terheltségét is.
Ezek közül az egyik módszer a távolságvektor alapú forgalomirányítás (distance vector routing).
Itt minden útválasztó rendelkezik egy táblázattal, amely két információt rendel minden egyes alhálózati csomóponthoz: a kimenetet és az ismert legrövidebb távolságot. A kimenet azt mondja meg, hogy az adott cél felé melyik porton keresztül juthatunk el a legrövidebben. A távolság értelemszerűen a cél távolsága. Tegyük fel, hogy most a távolság mértékegységének a késleltetési időt adjuk meg. Természetesen a távoság más is lehet, például az ugrások száma.

Az útválasztóknak táblázataikat rendszeres időközönként frissíteniük kell, ugyanis nem biztos, hogy az aktuálisan legrövidebb út egy óra múlva is a legrövidebb lesz (például egy útbaeső útválasztó annyira leterhelté válik, hogy azt megkerülve a csomagot hamarabb célba juttathatjuk). Ehhez azonban szükség van arra, hogy az útválasztók időnként információt cseréljenek egymás között.

A frissítési folyamatot a 2. *ábrán* illusztráltuk. Az ábra „a” részében láthatjuk magát az alhálózatot. Mi most a J router szemszögéből fogunk vizsgálódni.

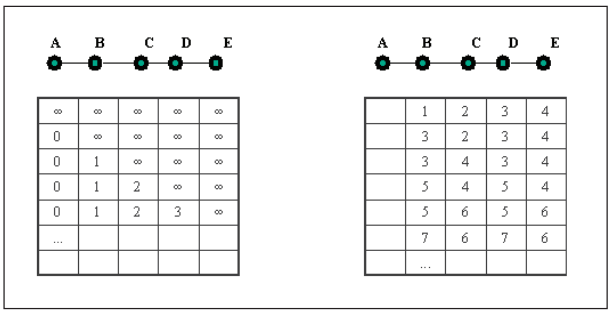
Az első és legfontosabb dolog, hogy minden útválasztó ismerje a szomszédai távolságát. Mivel a távolság most a késleltetés, amely időben folyamatosan változik, ezért az útválasztóknak szabályos időközönként méréseket kell végezniük. A késleltetést mérése úgy zajlik, hogy az útválasztó egy speciális csomagot, úgynevezett ECHO csomagot küld a szomszédjának. Ezzel a csomaggal az útválasztók nem csinálhatnak semmit, csupán beleírják az aktuális időt (úgynevezett időbélyeggel látják el), és visszaküldik a feladónak.

Az útválasztók időnként egy listát küldenek szomszédaiknak, amely tartalmazza az alhálózat összes pontjához tartozó becsült késleltetéseket. A 2*/b ábrán* feltüntettük, hogy a J útválasztó milyen – úgynevezett – késleltetési vektorokat kapott a szomszédaitól. A baloldali oszlop például az A útválasztótól érkező késleltetési vektor. Ha a vektor összes eleméhez hozzáadjuk a J és az A útválasztó közötti késleltetést (amely jelen esetben 8), akkor megkapjuk, hogy az A felé vezető kimeneten indulva mekkora utat kell megtennünk, hogy elérjük az alhálózat egyes pontait. Ezt a számítást el kell végeznünk a többi útválasztótól kapott késleltetési vektorra is. Ezután már csak ki kell választanunk az egyes alhálózati pontokhoz azokat a kimeneteket, amelyen keresztül a leggyorsabban eljuthatunk hozzájuk (2*/c ábra*). Nézzünk meg egy konkrét példát: a J és a G közötti útvonalat. Az A útválasztó azt állítja, hogy ő 18 ms alatt képes a csomagokat eljuttatni a G útválasztóhoz. Tudjuk, hogy az A tőlünk 8 ms-nyi késleltetésre van, tehát az A-n keresztül a G-t 18 + 8 = 26 ms alatt érhetjük el. Nézzük meg, mit mond a többi szomszédunk! Az I-n keresztül 31 + 10 = 41 ms, a H-n keresztül 6 + 12 = 18 ms, a K-n keresztül pedig 31 + 6 = 37 ms időbe telik, míg egy G felé tartó csomag célbaér. Ezek közül a H ajánlata volt a legkedvezőbb, azaz 18 ms, így az új forgalomirányító táblázatban a G ponthoz a H kimenetet rendeljük.

Az alhálózat többi pontjához tartozó új kimeneteket ugyanígy kaphatjuk meg.

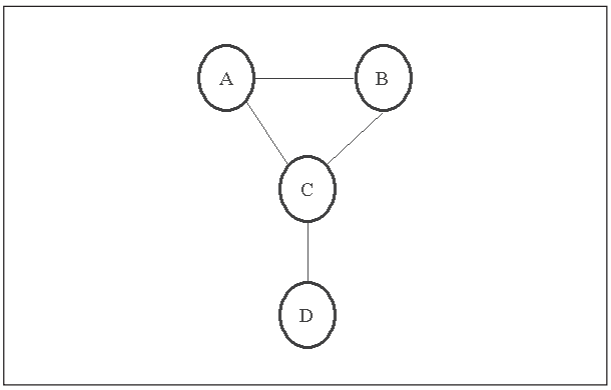
A végtelenig számolás problémája

Sajnos a távolságvektor alapú forgalomirányítás nem egy tökéletes algoritmus, a gyakorlati felhasználás során ugyanis adódig egy rendkívül kellemetlen probléma: míg a jó hírek (például egy korábban elhalálozott útválasztó dicsőséges feltámadása) gyorsan elterjednek az útválasztók között, addig a rosszakra az algoritmus hihetetlenül lassan reagál. Először nézzük meg, mi történik egy jó hír esetén. Az útválasztók közötti távolság legyen most az ugrások száma, továbbá tegyük fel azt is, hogy az útválasztók mindig szabályos időközönként, mindannyian egyszerre egyeztetnek egymással.



3. ábra A végtelenig számlálás problémája, a jó hírek (bal oldal) és a téves hírek (jobb oldal) következményei

A *3. ábrán* két táblázatot láthatunk, ezek közül a baloldali az, ami modellezi, hogy mi történik egy jó hír esetén. Legyen most a jó hír az eddig betegeskedő A útválasztó megjavulása. Kezdetben erről még egyik útválasztó sem tud semmit, ezért mindegyik végtelenre állítja az A-tól való távolságukat. Miután megtörténik az első útválasztók közötti információcsere, a B azt látja, hogy a baloldali szomszédja nulla távolságra van A-tól. Be is jegyzi a forgalomirányító táblázatába, hogy A tőle egy ugrásnyira van (ezt láthatjuk a baloldali táblázat második sorában). A többi útválasztó azonban erről még nem értesült. A következő cserénél a C útválasztó azt kapja, hogy a B egységnyi távolságra van A-tól. Mivel a C pontosan egy ugrásra van B-től, ezért bejegyzi a táblázatába, hogy tőle az A két ugrásnyira van. Láthatjuk, hogy a jó hír ugrás/csere sebességgel terjed, tehát egy olyan alhálózatban, ahol N a leghosszabb út hossza, ott N csere után minden útválasztó értesül a jó hírről.
Nézzük mi történik egy igazán rossz hír esetén, például az A elromlásakor (3/b ábra). Az első cserénél a B semmit sem kap A-tól. A C-től viszont értesül, hogy rajta keresztül létezik elindulva létezik egy 3 távolságú út. A B viszont nem tudhatja, hogy ez az út igazából olyan, hogy saját magán megy keresztül. Ezért a B-nek el kell fogadnia a C állítását, így beírja a táblázatába, hogy a C felé menve három lépésen belül el lehet jutni A-ba.
A második információcsere pillanatában a C azt kapja a szomszédaitól, hogy mindketten 3 távolságra vannak A-tól. Ekkor beírja a táblázatába, hogy az A-tól 4 ugrásra van, a két kimenet közül pedig kiválaszt egyet véletlenül (harmadik sor). A következő cserénél ugyanez a folyamat játszódik le. A probléma nyilvánvalóan az, hogy az útválasztók az A-tól való távolságukat minden cserénél csak eggyel növelik meg. Az algoritmus működése helyes, hiszen a cserék hatására a végeredmény konvergál a végtelenhez, csak ezt kivárni elég hosszú időbe telik. Ezt nevezzük a végtelenig számolás problémájának.
Hogy a gyakorlatban ez pontosan meddig is tart, az attól függ, hogy milyen számmal ábrázoljuk a végtelent.
Ha a távolság az ugrások száma, akkor annyival szerencsésebb a helyzet, hogy a végtelent megválaszthatjuk a leghosszabb út hosszától eggyel nagyobb értéknek. Ez az ugrások számára nézve egy biztos felső korlát. Ha a távolság a késleltetés, akkor elméletileg nincs ilyen felső korlát. Annyit tehetünk, hogy önkényesen kiválasztunk egy értéket, amelynél hosszabb késleltetés esetén az adott vonalat működésképtelennek tekintjük.



4. ábra Ilyen topológia esetén a megosztott láthatár módszere könnyen kudarcot vallhat

Megosztott láthatár

A ***megosztott láthatár (split horizon)*** egy népszerű ötlet a fenti probléma feloldására, ám ez is, mint a többi megoldási javaslat, bizonyos helyzetekben kudarcot vall. Sajnos idáig még nem született tökéletes megoldás, mindenesetre a megosztott láthatárt a gyakorlatban úgy-ahogy megállja a helyét, mivel csak néha hibázik.
Az ötlet az, hogy az útválasztó időnként hazudik a szomszédainak: egy X útválasztótól való távolságot végtelennek mond azon a vonalon, amelyre az X felé menő csomagokat irányítja. Csak az első hallásra bonyolult a dolog, a következő példából minden világos lesz. Nézzük megint a 3. ábrán lévő topológiát. A C útválasztó a D-nek elmondja az igazat az A-tól való távolságáról, viszont B-nek nem. Neki azt mondja, hogy végtelen a távolsága. Ugyanígy D is végtelent hazudik C-nek, E-vel viszont megosztja az A-tól való valódi távolságát.
Ez a megoldás valóban működik, ugyanis ha A tönkremegy, akkor B-nek C végtelen távolságot mond. Most már a rossz hír is ugrás/csere sebességgel terjed az alhálózaton.
Könnyű azonban olyan helyzetet kitalálni, amikor ismét szembesülnünk kell a végtelenig számlálás problémájával. Vegyük például a *4. ábrán* látható alhálózati topológiát, és tegyük fel, hogy a D útválasztó hirtelen meghibásodik. Az első csere pillanatában a C értesül erről, ugyanis D-től semmit sem kap, A-tól és B-től a megosztott láthatár következtében végtelen érték érkezik. Amint ez kiderül, C értesíti A-t és B-t, hogy a D elérhetetlen. Az A azonban tudja, hogy a B-nek van egy 2 hosszú útja D-be. A következő cserénél B meg arról értesül, hogy az A-nak létezik egy 3 hosszú útja D-be. Megint ugyanaz a helyzet, mint a 3. ábrán látott példa esetén.
Mindenesetre 1979-ig az internet elődje, az ***ARPANET*** ezt az algoritmust használta a forgalomirányításhoz. Lecserélését két ok indokolta: először is az algoritmus gyakran még így is túl lassan reagált a változásokra. Másodszor a távolság kiszámításához nem vették figyelembe a vonalak sávszélességét (habár ez a probléma egyszerűen megoldható lett volna). Így a távolságvektor alapú algoritmust felváltotta a kapcsolatállapot alapú forgalomirányító algoritmus, amellyel sorozatunk következő részében foglalkozunk.

Garzó András
garzo@interware.hu

Főzzünk Linuxszal – szemrevaló panelek

Ha már beleuntunk a jól működő de unalmas felhasználói felületekbe, öltöztessük fel asztalunkat csinos panelekkel és 3D ablakváltóval.

François? Kicsit zöldnek látszol. Mi a baj? Ah, szóval a 3D asztalváltóval játszottál és elfogott a tengeribetegség. A te állapotodban azt hiszem jobb ha a hagyományos asztali lapváltóknál maradsz. Nem, egyáltalán nem, ***mon ami***, nem viccelek. Az ilyen jóravaló földhözragadt pincérnek mint te, igazán nem való, hogy a dolgok össze-vissza keringjenek a térben. Szedd össze magad, ***François***. Ugye nem szeretnénk, hogy ráejtsd vendégeinkre a bort felszolgálás közben. Vendégeinkről beszélve, éppen meg is érkeztek. Üdvözlet, ***mes amis Chez Marcelnél***, a világ legjobb ***Linux*** francia éttermében és a világ legjobb borospincéjében. Helyezzék magukat kényelembe. Azonnal küldöm hűséges pincéremet a borospincébe. Készülj, ***François***. Nézzük csak... a 2003-as ***Casillero del Diablo Chilei Chardonnay*** kitűnő lenne ezzel a menüvel – friss körte és zöldalma ízzel és éppen a megfelelő savassággal, ***mes amis***. Szívem szerint megsürgetnélek ***François***, de inkább vigyázz a feljövetellel!

A számunkra természetes megoldásokkal szemben, sok programozó és felhasználó inkább más megoldásokat keresne a hagyományos panel, lapozó és rendszerasztal helyett. Az az érdekes mindebben, hogy e panel helyettesítéseken (vagy fejlesztéseken) végzett kemény munka szemrevaló 3D hatások formájában nyilvánul meg. S éppen ez az ami ***François*** lábait kicsit bizonytalanná tette.

Az egyik alternatív panel ***Stephano KXDocker Projekje***, amely némiképp a ***Mac OS X Docker*** felületre hajaz, de ahogy ***Stephano*** állítja „annál is hatékonyabb". A létrejött hatás következtében a rendszerünkön alul végig különféle alkalmazásokat (a program indító menüket is ideértve) jelképező ikonok sora jelenik meg. Az egeret végigfuttatva ezeken az ikonokon, valamiféle ikonhullámhoz hasonlatos hatást kelthetünk a képernyő alján (*1. ábra*).

A ***KXDocker*** életre leheléséhez először is szerezzünk be egy másolatot a projekt honlapjáról (lásd a hálózati forrásokat). A weblapon előre fordított csomagokat találunk meglepően sok nagyobb terjesztéshez. Ha a rendszerünk nem szerepelne a listában, a forrást is megtaláljuk.

A letöltőlapon egy erőforrás csomagot is találunk. Ez ugyan nem szükséges a legújabb változathoz, azonban van benne néhány további tématámogatás, tehát lehet, hogy ezt is érdemes feltennünk (egyszerűen futtassuk a `install.sh` parancsfájlt). Forrásból fordítani a szokásos tömöríts-és-fordíts ötlépéses módszerrel tudunk:

1. ábra Ez a KDE panel hulllámoztatja az ikonjainkat

```
tar -xjvf kxdocker-0.23.tar.bz2
cd kxdocker-0.23
./configure --prefix=/usr
make
su -c "make install"
```

A szokásos `./configure` lépést most kibővítettem a `prefix` kapcsolóval, hiszen a `KXDocker` programot érdemes ugyanabba a fába helyezni ahol a ***KDE*** telepítésünk található.

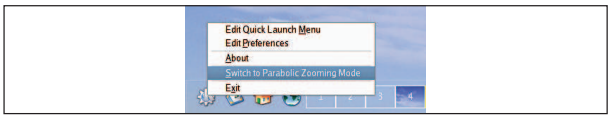
A ***KXDocker*** használatához egyszerűen csak futtassuk a `kxdocker`-t. A tálca a képernyő alján jeleik meg. Valószínűleg nem árt, ha a ***KDE Kicker*** panelt eltesszük az útból (egyelőre húzzuk felülre). Bár a ***KXDocker*** eredetileg a ***KDE kicker*** helyettesítésére tervezték, nagyon szépen fut mellette is. Tulajdonképpen a ***KXDocker*** még a rendszertálcába is beágyazható, ahonnan egyetlen kattintással elővarázsolhatjuk.

Az alapértelmezett műveletek (ikonok témák) megváltoztatásához jobb gombbal a panelre kattintva válasszuk a ***Configurator***-t (ugyanezt elérhetjük a rendszertálca ikonján jobb gombbal kattintva). A ***configurator*** fülekkel ellátott űrlap, ahol számtalan dolgot módosíthatunk, és az asztalt tetszés szerint az ízlésünkhöz igazíthatjuk. Az egyik beállítás amit azonnal át is állíthatunk, a ***Window*** fül alatt található ***Auto send to background*** néven. Ezt átállítva a panel nem tűnik el automatikusan amikor egy alkalmazást (például szövegszerkesztőt) futtatunk. A változtatások elvégzése után kattintsunk a ***Save*** ikonra és rendeljünk egy nevet a beállításunkhoz. Amikor megkérdezi, hogy szeretnénk-e hogy a ***KXDocker*** indulásakor automatikusan elinduljon, fogadjuk el.

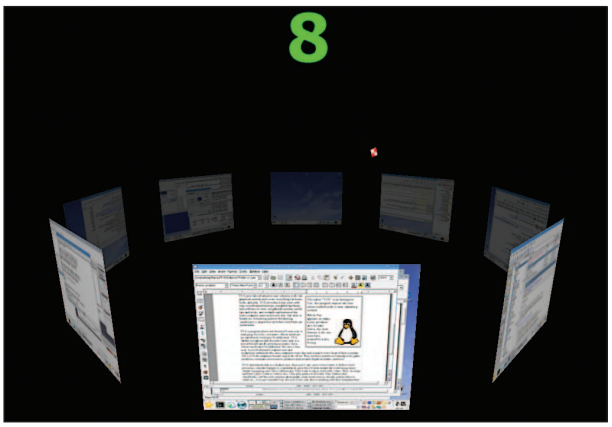
Amennyiben az asztalélményünk fokozása érdekesen hangzik, ***mes amis***, ne álljunk itt meg. Egy másik, figyelmünkre érdemes projekt a ***KSmoothDock*** csapat ***KsmoothDock***-ja. A ***KSmoothDock*** két különböző nagyítási módban működik. Az alapértelmezett neve ***normal*** nagyítási mód. Ahogy az ikonokat az új panelen mozgatjuk az ikonok mérete megnő, hogy jobban láthassuk őket (*2. ábra*).

2. ábra KSmoothDock normál nagyítási módban

3. ábra Mac OS X a látványt ismerősnek találhatják



4. ábra Jobb gombbal váltunk nagyítási módot



5. ábra A 3D-asztal alapértelmezett megoldása a carousel

De ez csak a kezdete és a legkezdetlegesebb módja a *KsmoothDock* beállításainknak. Hamarosan megnézzük a többit is, de előbb indítsuk be a *KsmoothDock*-ot, amihez egy másolatra lesz szükségünk. Bár a *KSmoothDock* hivatalos lapja a *SourceForge*, a legjobb és legfrissebb csomagot inkább *KDE-Look* weblapján érdemes beszereznünk (lásd a forrásokat). Innen előrefordított állományokat tölthetünk le néhány terjesztéshez. A forrás úgyszintén elérhető, amit bármilyen *KDE 3.2* vagy későbbi változathoz feltehetünk. A módszer ismét a jó öreg ötlépéses tömörít és fordít megoldás:

```
tar -xzvf ksmoothdock-3.5.1.tar.gz
cd ksmoothdock-3.5.1
./configure --prefix=/usr
make
su -c “make install”
```

A programot a `ksmoothdock` paranccsal indíthatjuk. Indításkor egy ablak jelenik meg, miszerint a *KDE* alapértelmezett *Kicker* paneljét mozgassuk felülre, el az útból. Az ablak mindjárt fel is ajánlja, hogy ezt megteszi helyettünk.
A `ksmoothdock` program futása közben a virtuális képernyők lapozóját kivéve minden ikon méretűvé zsugorodik, még a futó folyamatok is. Az ikonok az alkalmazáshoz tartozó alapértelmezett ikonok lesznek.
A második módot *parabolic* nagyításnak nevezik és már jobban hasonlít a *KXDocker* által létrehozott jelenséghez. *Normál* módban, a virtuális asztalokat számozott négyzetek jelképezik, amik nem nagyítódnak ki. *Parabolic* módban mindez megváltozik amint ezt a *3. ábrán* láthatjuk.

A *parabolic* módba váltáshoz jobb gombbal bökjünk az tálca programindítójára (egészen balra) és válasszuk a *Switch to Parabolic Zooming Mode* pontot (*4. ábra*). Az ilyesfajta módváltáshoz azonban ki kell lépnünk a programból, majd újra kell indítanunk, hogy a változásainknak érvényt szerezzünk. Ugyanez igaz a normál módra, ha esetleg túl mozgalmasnak találnánk a nagyíttatást és mégis vissza szeretnénk váltani.
A 4. ábrán bemutatott menü két másik érdekes elemet is tartalmaz. A legfelső elemmel megváltoztathatjuk a *Quick Launch* menüt, azaz a virtuális asztalok jobboldalán található négy alapértelmezett menüt. A megnyíló *Konqueror* ablakban hivatkozásokat hozhatunk létre az alkalmazásra.
A második a *Preferences* űrlap. A *Preferences* menüben kiválaszthatjuk melyik elemek legyenek láthatóak a tálcán (azaz például legyen-e óra) illetve a tálca ikonok megjelenjenek-e vagy sem. Másik érdekes beállítási lehetőség, az áttetszőségi szint megadása, mellyel beállíthatjuk, hogy az asztal háttere milyen mértékben tűnjön át a tálcán.
Valamennyi esetben az egytelen dolog ami többé kevésbé azonosa marad mindig, a lapozó és a virtuális asztalok – itt semmi különösen látványosat nem kapunk, eltekintve az egyszerű megjelenítési ikonnagyítástól. Ezzel kapcsolatban egy igazán ízletes desszertet szándékozok felkínálni amivel mindent kihozhatunk a rendszerteljesítményből, és az egyik leglátványosabb hatásokat hozhatjuk létre amit csak valaha láttam. *Brad Wasson* 3D-asztaláról beszélek, arról az *OpenGL* programtól, amely látványos módon teszi lehetővé, hogy az egyik virtuális asztalról a másikra váltsunk. Természetesen egyértelműen szükségünk van egy 3D gyorsítással rendelkező kártyára az ilyen mutatványhoz. Amikor a program elindul, a képernyő 3D módba vált. A jelenlegi virtuális asztalunk eltűnik, és az egész dolog úgy tűnik elő, mintha minden ablakunk valahol az űrben lebegne. Elképesztően jól néz ki, mindenképpen érdemes kipróbálni. Alapértelmezés szerint a 3D asztal kezdeti nézete a *carousel*, ahol a virtuális asztalink körben helyezkednek el (*5. ábra*). Egyik asztalról a másikra ugorni a jobb és bal kurzorbillentyűk segítségével lehet. Miután kiválasztottuk az asztalt, nyomjuk meg a szóköz vagy az ENTER billentyűt. A kiválasztott virtuális asztal felnagyítódik, és a képernyő normál képernyővé alakul. Nagyon szuper! És jó móka. Sőt még hasznos is.
A *3D-Desktop* élmény kipróbáláshoz először is fel kell telepítenünk a rendszerünkre, úgyhogy lépjünk be a *SourceForge* lapra és töltsünk le egy másolatot (lásd a forrásokat). A néhány bináris csomag mellett (*SuSE* és *Red Hat*) mellett letölthetjük a forráscsomagokat és a forrás RPM-et. A hozzájárulók lapján valószínűleg találunk saját terjesztésünkhöz is csomagot, de ha forrásból kell lefordítanunk, az sem jelenthet különösebb nehézséget. Szükségünk lesz a *Mesa GLU* és *Imlib2* fejlesztési könyvtárakra, de ezen túlmenően ez is klasszikus példája a szokásos tömörítsd és fordítsd öt lépéses folyamatnak:

```
tar -xzvf 3ddesktop-0.2.7.tar.gz
cd 3ddesktop-0.2.7
./configure
make
su -c “make install”
```

A program indítása mindössze egy `3ddesk` parancs kiadásából áll. Ugyanakkor első ízben a programot az `--acquire` kapcsolóval kell lefuttatnunk. A program indítása ezzel a módszerrel két célt szolgál. Az egyik, hogy ellenőrizzük a program kiszolgáló része (`3ddeskd`) fut-e és elindítsuk, ha nem. A másik, hogy direkt módon begyűjtsük valamennyi meglévő virtuális asztalunk képeit. A legtöbb ember négy virtuális asztalt futtat. Én nyolcat. A folyamat csak egy-két másodpercet vesz igénybe. Rögtön ezután létrejön a mágia és a 3D váltó már fut is.

Miután kiválasztottunk egy virtuális asztalt és visszatértünk a munkához, legközelebb ismét le kell futtatnunk a `3ddesk` programot. Ennek kiváltására rendeljünk egy nem használt funkcióbillentyűt a használatához. Én ***KDE*** alatt az F2-t használtam, így az F2 leütésével egyetlen gombnyomással 3D-asztali nézetbe válthatok. Más asztali környezetekben másképp kell beállítani, de ***KDE*** alatt a következőket kell tennünk (a 3D-asztal weblapján találunk ötleteket más rendszerekhez is).

Jobb gombbal bökjünk a nagy ***K***-ra (alkalmazásindítóra) és válasszuk a ***Menu Editor (menüszerkesztő)*** pontot. Amikor a menüszerkesztő ablak megjelenik, gyalogoljunk le oldalt a kiválasztott alkalmazásig (Lehet hogy új 3D-asztal menüelemet kell felvennünk, a ***File*** majd ***New Item*** gombokkal). Kattintsunk a 3D-asztalhoz-hoz tartozó bejegyzésre és pillantsunk az ablak jobb oldalának alsó része felé. Látjuk a ***Current shortcut key (aktuális gyorsbillentyű)*** gombot? Valószínűleg a ***None (semmilyen)*** szót találjuk itt. Kattintsunk a gombra. Az ablak most egy billentyű lenyomására vár. Nyomjuk le az F2-t (avagy tetszés szerinti más kombinációt), és kattintsunk az ***Apply (Alkalmaz)*** gombra. Most már bezárhatjuk a menüszerkesztőt.

Játszhatunk egy kicsit a parancssori kapcsolókkal is. Bár személy szerint az alapértelmezett ***carousel*** nézet a kedvencem, más érdekes módok (soros, fordítgatós és egyebek) is beállíthatóak. Gyerekkori nosztalgiaként próbáljuk ki a ***viewmaster*** módot a `3ddesk --mode=viewmaster` paranccsal. További példákért gépeljük be a `3ddesk --help` parancsot.

Úgy tűnik, ***mes amis***, a záróra ismét lesújtott ránk. Azonban biztos vagyok benne, hogy François szívesen feltölti még egyszer vendégeink poharát. ***Merci***, François. Meg kell vallanom ez a bor különösen kiváló. Segít ráébrednem, hogy a sima, lapos, valódi asztal az amire igazán szükségünk van – jófajta erős lap, amin a borospoharaink megállnak, ***non***? Addig is, ***mes amis***, igyunk egymás egészségére.

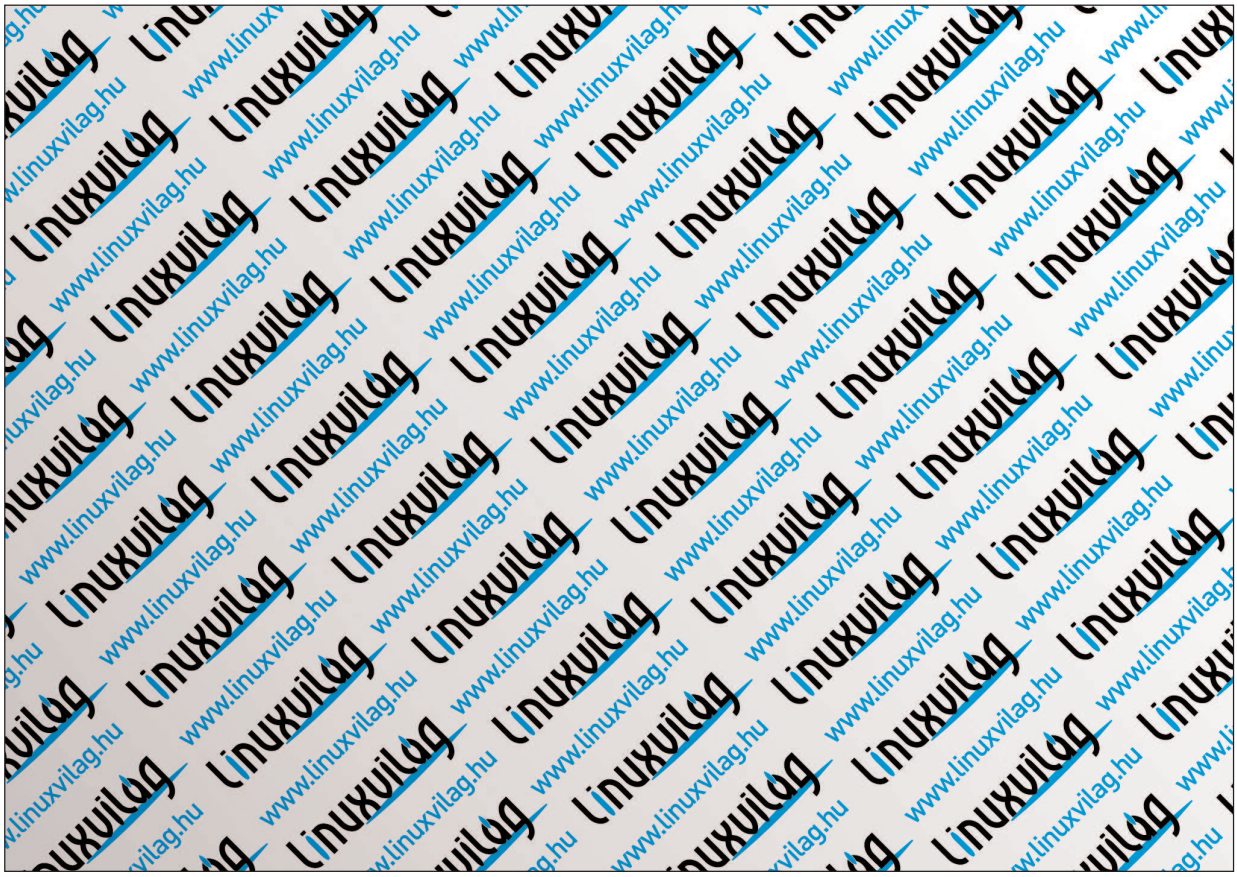
A ***votre santé! Bon appétit!***

Linux Journal 2005. február, 130. szám

A cikkhez tartozó források:

➲ ***www.linuxjournal.com/article/7921***

Marcel Gagné (mggagne@salmar.com)
Mississaguában, Ontario államban él.
Ő a szerzője a Kiskapu kiadásában tavaly szeptemberben megjelent Linux-rendszerfelügyelet (ISBN 96-9301-40) című könyvnek.



Munka- és szolgálati viszony II. Az egyes speciális kérdések

Személyhez fűződő jogok.

Ezeket, a szerzőtől alapvetően elválaszthatatlan jogokat a munkaviszonyból kifolyólag általában véve a munkáltató érvényesíti. Ezen változást a német Kommentár azzal magyarázza, hogy a mű alig tekinthető egyedi alkotói teljesítménynek, és a mű és alkotója közötti szellemi és gondolati kapcsolat gyenge.

Ami a nyilvánosságra hozatalt illeti, a szerző legkésőbb a mű elkészültével és átadásával lemond a nyilvánosságra hozatal jogáról,[1] egyes szerzői jogászok szerint a szerző munkaviszonyba állásával már eleve lemond erről a jogáról.

A munkáltatót megilleti azon jogosultság, hogy a művet copyright jelzéssel (©) lássa el, és ezáltal, mint jogtulajdonos a szerző mellett (vagy helyett) a szerzői minőséget bitorló harmadik ellen is fellépjen. A szerzőként való feltüntetésről a szerző vagy a munkaszerződés keretei között mond le, vagy néha az is elég, ha tisztában van a vállalat gyakorlatával. Ezen eljárás magyarázatául a szoftverháznak a neve alatti terjesztéshez fűződő jogos érdeke szolgál. Ugyanakkor felmerül a kérdés, hogy ez a szerzőként való feltüntetés, miután a szoftverház követelései kielégítést nyertek – azaz a dobozon, és a CD külsején valóban csak a copyright jogosult jelenik meg – nem igényelhető-e mégis, hiszen voltaképpen alapvető, személyhez fűződő jogosultságról beszélünk.

A szerzők feltüntetésének gyakorlott módja a forráskódban való megjelölésük, ám ezáltal nem érik el a kívánt célt, hiszen, nem válnak ismertté és népszerűvé, mert a forráskód az esetek nagy részében nem ismerhető meg. Hiszen azt fő szabály szerint – ám vitatható jogszerűséggel – egyáltalán nem adják át, előfordulhat az a jogilag inkább helyénvaló megoldás is, hogy a kódot szigorú feltételek mellett letétbe helyezik. A legoptimálisabb eset mégis a nyílt forráskódú szoftvereké, melyek kódja – as az abban megjelenő nagy számú szerző egyaránt megismerhető.

A másik megoldás a szerzők kézikönyvben való felsorolása, az ezzel kapcsolatos, aggályok alapja, hogy nem a szoftverhez tartozó dokumentációról, hanem egy külön, önálló szerzői jogi védelmet élvező műről van szó, így a közvetlen kontaktus az alkotó és a mű között elvész.

Úgy tűnhet, az egyetlen, valóban a szerző kezében maradt jogosultság a mű egységének védelme, de ennek is kizárólagosan azon része, mely a visszaélésszerű megváltoztatások – rasszista, illegális tartalommal való megtöltése – ellen való fellépésre vonatkozik. Ugyanakkor más jellegű módosításokat engedélyeznie kell a szerzőnek vagy akár az engedélye nélkül is kivitelezhetnek, és amennyiben ezekkel nem ért egyet legfeljebb nevének elhallgatását kérheti.[2]

A szerző a forráskód kiadását nem kérheti, hiszen az már átadástól – egyes nézetek szerint megalkotástól – a munkáltató „műhelytitkának" minősül. Pont ezért jogosult is a munkáltató a forráskód megváltoztatását kérni a szerzőtől vagy ezzel valamely más programozót megbízni.

Ugyanakkor egy ehhez hasonló érdekes eset merült fel Németországban, mikor a munkavállaló korábbi munkahelyének mintegy végkielégítéseként – egy bizonyos szoftverre vonatkozó – vagyoni jogosultságokat nyert el. Rendelkezési joga folytán ezt a szoftvert az új munkahelyen ellenszolgáltatás nélkül használni engedte, és mikor munkaviszonya megszűnt, szerette volna a programot magával vinni, de a munkáltatója ehhez nem járult hozzá, mert a szoftver használatát úgy tekintette mint a munkavállaló munkaköri kötelességéből származó eredményt, habár a munkavállaló a szoftvert nem maga és főként nem ezen a munkahelyén fejlesztette. Az eset megoldásának – mindamellett, hogy az eljáró bíróság szerepét a munkajogi bíróság helyett a rendes bíróságnak kellett volna ellátnia – az tűnik, hogy az ingyenes felhasználást abban az esetben is csak a munkaviszony fennállásának idejére szabták meg, amennyiben a felhasználás időtartamáról külön megegyezés a felek között nem történt.[3]

A visszahívás joga a szoftver gyártóját illeti, hiszen ez – egy igen nagy horderejű – gazdasági döntésnek minősül, a szoftver szerző meggyőződésének megváltozása a témával foglalkozó többség meglátása szerint elképzelhetetlen.

[1] Szjt. 30.§ (5)
[2] Szjt. 30.§ (5)
[3] BAG, Urteil v. 17. 9. 1998, Hessen in: NJW 1999, 1275

Vagyoni jogok

A vagyoni jogoknak, mint már néhány számmal ezelőtt említettük, mindig van jogosultja (aki a főbb felhasználási lehetőségekről rendelkezhet) jelen esetben ez elsősorban a szerzőt, míg másodsorban a munkáltatót.

A régi és az új szerzői jogi törvény legfontosabb változása a szerzőket megillető „megfelelő díj” sorsa, amely abban az esetben járt részükre, ha a munkáltató a felhasználásra másnak engedélyt ad vagy a művel kapcsolatos vagyoni jogokat másra átruházza.[4] Ahogy a korábbi bírósági döntések is mutatják,[5] a szerzőket szoftverek esetében is megillette bizonyos – többnyire a szakértők által javasolt – díjazás. Az 1999 előtt irányadó rendelkezések szerint[6] a munkáltató által harmadik személlyel kötött felhasználási szerződésbe foglalt a szerzői díj – szoftver esetében nevesítetten – 10-30%-ban a szerzőt illette meg. A szoftvert esetleges átdolgozásainak költségei azonban a munkáltató ráfordításának minősültek, így azokra tekintettel a szerzőt megillető díj összegét 10% alatt is meg lehet állapítani.[7] A szerzői jogi törvény jelenlegi változata azonban nem teszi lehetővé az említett megoldást.[8] Ennek a törvényalkotó által adott magyarázata, hogy a szoftvergyártók oldalán tetemes beruházást igényel a szoftverek előállítása, és ezen befektetés megtérülésének kerékkötője lenne, amennyiben a szoftverkereskedelem lényegi elemét érintve a befolyó díjból a szerzőt is részesíteniük kellene. Ehhez kapcsolódóan érdemes megemlíteni – a magyar joggyakorlat híján – két érdekes német döntést, a Wetterführungspläne I-II.[9] döntéseket. Itt a vita abból fakadt, hogy egy munkavállaló munkatársával 1979 és 1992 között egy speciális program fejlesztésével foglalkozott. Az első döntésében a Szenátus kimondta, hogy a munkaköri kötelessége körében alkotott szoftver nem jogosítja fel a munkavállalót a munkabért meghaladó semmiféle követelésre. Alig egy évvel később a Szövetségi Bíróság még egy döntést hozott az ügyben, melynek keretében új eljárást rendelt el, felhívva a figyelmet a „bestseller paragrafus” figyelembevételére, azaz annak mérlegelésére, hogy a munkáltatónál keletkezett vagyoni előny és a munkavállaló számára munkabérként kifizetett díjazás arányossága tekintettel a program nem várt népszerűségre, feltűnően nagy értékkülönbséget generált a vagyoni jogokat a törvény szövegéből adódóan, mintegy „törvényi licenc” által megszerző társaság és a program alkotója számára juttatott összeg között.

A hazai szabályozásnak ugyancsak szépséghibája a munkaviszonyban alkotott szoftverekkel kapcsolatos „megfelelő díj” mértékének megítélése. A korábbi szabályozás az ismertetett módon engedélyezte ugyan, a többi munkaviszonyban alkotott műhöz hasonlóan a harmadik fél felé újabb és újabb felhasználási jogok biztosításával a szerző részesedését is, ám ez a már említett okból kikopott. Ellenben a magyar szabályozásban is megjelent a már említett bestseller paragrafus kitétel, mit eredendően a könyvekre vonatkoztatottan kívántak alkalmazni, ám a kor (mint annyiszor) túlmutatott a jogalkotói szándékon. Megteremtve ezáltal egy új, mintegy „kompenzáló” területet, amely voltaképpen kiskaput biztosítva a szoftverszerzők részére, mely alapján a vagyoni jogok átruházásáról szóló (akár törvényi, akár a felek megegyezésén nyugvó) felhasználási szerződés megváltoztatását – különböző feltételek együttállása esetében – kérhetik. A polgári jogból jól ismert feltűnő értékaránytalanság speciális változatával állunk szemben, hiszen itt a mű tényleges piaci értékét először a forgalomba hozatalt követően lehet megállapítani. Sajnos, az említett esettel kapcsolatban magyar döntés még nem született, és tudtommal eddig ilyen pert nem is indította.

Dr. Dudás Ágnes (dudas.agnes@abend.hu) ügyvédjelölt, az FSF egyik aktivistája. 2004-ben végzett az ELTE Jogtudományi Karán. Szakdolgozatát a szoftverek szerzői jogi védelméről írta, a 2003-as évet pedig e terület kutatásával a berlini Humboldt Egyetemen töltötte.

[4] Szjt. 30.§ (3)-(4)

[5] például: BH 1991.231. Fővárosi Bíróság 8.P.25.250/92

[6] „Vhr. 12. § (1) Ha a munkáltató az Szjt. 14. §-ában biztosított felhasználási jogának gyakorlása során a mure harmadik személlyel köt felhasználási szerződést, a szerzői díj összegének – a munkáltató döntése szerint – 60-80 százaléka, szoftver esetében 10-30 százaléka a szerzőt illeti meg, amit a munkáltató a szerzői díj felvételétől számított 8 napon belül köteles a szerző részére kifizetni. Szoftver esetében a kifizetés határidejét a munkáltató – a munkaszerződésben vagy más módon – ettől eltérően is meghatározhatja, illetve feltételhez kötheti. Amennyiben a műre harmadik személlyel felhasználási szerződés kötése a munkáltató feladatkörébe tartozik, a munkáltató a mű szerzőjének díját – a mű alkotásával kapcsolatos ráfordításokra figyelemmel – a szerzői díj 60 százalékánál, szoftver esetében 10 százalékánál alacsonyabb mértékben is meghatározhatja.” 9/1969. (XII. 29.) MM rendelet

[7] Ezzel kapcsolatos döntés: BH 1984.269.

[8] „A szerző munkaviszonyból folyó kötelessége teljesítéseként elkészített szoftverre a 30.§ (3)-(4) bekezdésben foglalt rendelkezések nem vonatkoznak.” Szjt. 58.§ (4)

[9] BGH, Urteil v. 24. 10. 2000 – X ZR 72/98 Düsseldorf in: NJW-RR 2001, 626 illetve BGH Urteil 23. 10. 2001X ZR 72/98 Düsseldorf in: NJW-RR 2002, 339